REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public Reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimates or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188,) Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE **31 March 1998**	3. REPORT TYPE AND DATES COVERED **Technical Report**

4. TITLE AND SUBTITLE **Logic Programs, Well-orderings and Forward Chaining**	5. FUNDING NUMBERS **DAAH04-96-1-0341**
6. AUTHOR(S) **W.V. Marek, A. Nerode and J.B. Remmel**	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Regents of the University of California** **c/o Sponsored Projects Office** **336 Sproul Hall** **Berkeley, CA 94720-5940**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING / MONITORING AGENCY REPORT NUMBER ARO 35873.65-MA-MUR

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by the documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

We invesitgate the construction of stable models of general propositional logic programs. We show that a forward-chaining technique, supplemented by a properly chosen safeguards can be used to construct stable models of logic programs. Moreover, the proposed method has the advantage that if a program has no stable model, the result of the construction is a stable model of a subprogram. Further, in such a case the proposed method isolates the inconsistency of the program, that is it points to the part of the program responsible for the inconsistency. The results of computations are called stable submodels. We prove that every stable model of a program is a stable submodel. We investigate the complexity issues associated with stable submodels.

14. SUBJECT TERMS **complexity, stable model, forward chaining**	15. NUMBER OF PAGES **50**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

19980519 051

Center for Foundations of Intelligent Systems

DTIC QUALITY INSPECTED 2

CORNELL
UNIVERSITY

625 Rhodes Hall, Ithaca, NY 14853 (607) 255-8005

Technical Report
98-05

Logic Programs, Well-orderings and Forward Chaining

V. W. MAREK, A. NERODE AND J. B. REMMEL

March 1998

To appear in Annals of Pure and Applied Logic

Logic Programs, Well-orderings, and Forward Chaining

V.W. Marek [a,1] A. Nerode [b,2] J.B. Remmel [c,3]

[a] *Department of Computer Science*
University Kentucky, Lexington, KY 40506-0027.

[b] *Mathematical Sciences Institute,*
Cornell University, Ithaca, NY 14853.

[c] *Department of Mathematics,*
University of California at San Diego, La Jolla, CA 92903.

We investigate the construction of stable models of general propositional logic programs. We show that a forward-chaining technique, supplemented by a properly chosen safeguards can be used to construct stable models of logic programs. Moreover, the proposed method has the advantage that if a program has no stable model, the result of the construction is a stable model of a subprogram. Further, in such a case the proposed method "isolates the inconsistency" of the program, that is it points to the part of the program responsible for the inconsistency. The results of computations are called *stable submodels*. We prove that every stable model of a program is a stable submodel. We investigate the complexity issues associated with stable submodels. The number of steps required to construct a stable submodel is polynomial in the sum of the lengths of the rules of the program. In the infinite case the outputs of the forward chaining procedure have much simpler complexity than those for general stable models. We show how to incorporate other techniques for finding models (e.g. Fitting operator, Van Gelder-Ross-Schlipf operator) into our construction.

1 Introduction and Motivation

One of the problems which motivated this paper is how do we deal with inconsistent information. For example, suppose that we want to develop an expert

[1] Research partially supported by NSF grant IRI-9400568.
[2] Research partially supported by USARO MURI DAAH-04-96-10341, Center for Foundations of Intelligent Systems at Cornell University.
[3] Research partially supported by NSF grant DMS-93064270.

system using logic programming with negation as failure. It may be the case that the knowledge engineer gathers facts, i.e. clauses of the form $p \leftarrow$, rules without exceptions, i.e. clauses of the form $p \leftarrow q_1, \ldots q_n$, and rules with exception or rules of thumb, i.e. clauses of the form $p \leftarrow q_1, \ldots q_n, \neg r_1, \ldots, \neg r_m$, from several experts. One problem is that the resulting program may be inconsistent in the sense that the program has no stable model. That is, the experts may not be consistent. The question then becomes how can we eliminate some of the clauses so that we can get a consistent program. That is, at a minimum, we would like to select a subprogram of the original program which has a stable model. Various schemes have been proposed in the literature to do this [GS92,GS93,KL89]. For example, we may throw away the rules which came from what we feel are the most unreliable experts until we get a consistent program. However even in the case when the knowledge engineer consults only a single expert, the rules that the knowledge engineer produces may be inconsistent because the rules that he or she abstracted are not specific enough or simply because the expert did not give us a consistent set of rules.

The above scenario is one practical reason that we would desire some procedure to construct, for a given program which has no stable model, a maximal subprogram that does have a stable model. Another practical reason occurs when we are using a logic program to control a plant in real time, see [KN93a] for examples. In this case, the program may have a stable model but that stable model may be very complicated and we do not have enough time to compute the full stable model. It has been shown [MT91] that the problem of determining whether a finite propositional logic program has a stable model is NP-complete. Moreover, the authors have shown [MNR92a] that there are finite predicate logic programs which have stable models but which have no stable models which are hyperarithmetic so that there is no possible hope that one could compute a stable model of the program no matter how much time one has. Thus if there are time problems, one may be satisfied by a procedure which would construct a subprogram of the original program and a stable model of the subprogram as long as both the subprogram and stable model of the subprogram can be computed rapidly, in fact, in polynomial time.

Indeed some see as a general problem with the stable model semantics the fact that there are many programs which have no stable models. For example, if we have any program P and p is new statement letter, the program P plus the clause $p \leftarrow \neg p$ has no stable model even if the original program P has a stable model. Thus a single superfluous clause which may have nothing to do with the rest of the program may completely destroy the possibility of the program possessing a stable model. This is one of the reasons that researchers have looked for alternatives to the stable model semantics such as the well-founded semantics [VGRS91].

In this paper, we shall present a basic Forward Chaining type construction which can be applied to any general logic program. The input of the construction will be any well-ordering of the non-Horn clauses of the program. The construction will then output a subprogram of the original program and a stable model of the subprogram. It will be the case that for any stable model M of the original program P, there will be a suitable ordering of the non-Horn clauses of the program so that the subprogram produced by our construction is just P itself and the stable model of subprogram produced by our construction will be M. Thus all stable models of the original program will be constructed by our Forward Chaining construction for suitable orderings. Moreover, we shall show that for finite propositional logic programs, our construction will run in polynomial time. That is, we shall prove that our Forward Chaining construction runs in order of the square of the length of the program.

In fact, a basic Forward Chaining (FC) construction can be applied to any nonmonotonic rule system. In [MNR90,MNR92c], it was shown that nonmonotonic rule systems capture all the essential features of many nonmonotonic reasoning formalisms, including general logic programming with classical negation [GL90], Reiter's default logic [Rei80], modal nonmonotonic logics of McDermott [McD82] and truth maintenance systems of [Doy79,RDB89]. In the setting of nonmonotonic rule systems, one can give general proofs for many of the basic theorems about such nonmonotonic reasoning formalisms. Our Forward Chaining construction can thus be applied to any of these formalisms. This can be done by translating a nonmonotonic system as above into a nonmonotonic rule system of [MNR90] and then writing an interpreter of such a rule system within logic programming with stable semantics. The Forward Chaining construction will then take any well-ordering $\prec$ of the nonmonotonic clauses of a nonmonotonic system $\mathcal{S} = \langle U, N \rangle$ and produce a subset $C^{\prec}$ of nonmonotonic rules of $\mathcal{S}$ and a set $D^{\prec} \subseteq U$ which will be an extension of the system $\langle U, C^{\prec} \rangle$. Thus the results of this paper apply to all the systems mentioned above.

We shall see that any stable model M of P can be produced via our Forward Chaining construction for some well-ordering $\prec$, i.e. every stable model of P is a stable submodel of P. In the case where our original program P is inconsistent in the sense that P has no stable models, we can view our Forward Chaining construction as a way of extracting a maximal consistent subset of clauses $C^{\prec} \subseteq P$ such that the system $C^{\prec}$ has stable model. As outlined above, this feature of the Forward Chaining construction has a number of potential applications. In particular, in the construction of expert systems, one often consults several experts and the rules of different experts may conflict. Thus the designer of the expert system is left with the task of extracting a consistent set of rules from the rules supplied by different experts. We shall see that our Forward Chaining construction is ideally suited to this task for it allows us to favor the rules of one expert over another by the simple process of placing the rules of our favored expert earlier in the list. Our results apply equally well to the construction of extensions of default theories, answer sets for extended logic programs, expansions of modal nonmonotonic theories or extensions of truth maintenance systems.

We shall also analyze the complexity of our Forward Chaining construction. We shall show that for general recursive program, we can always produce a stable submodel which is r.e. in the jump of the empty set, $\mathbf{0}'$. Note that in [MNR95], the authors constructed a recursive program system P such that P has stable models but no hyperarithmetic stable models. Thus we are always guaranteed that a recursive program has a stable submodel which occurs at a relatively low level in the arithmetic hierarchy where no such guarantee can be made for stable models of recursive programs even when such programs have stable models. More importantly, we shall show that for finite programs, we can always find a stable submodel and its corresponding subprogram in polynomial time. Thus our Forward Chaining construction has potential applications for real time systems.

The outline of this paper is as follows. In Section 2 we shall briefly review the basic concepts of logic programming. In Section 3 we shall introduce our Forward Chaining construction and prove several basic results about the construction. In Section 4, we shall introduce recursive programs and recall some basic results about such programs proved in [MNR92c]. Then we shall prove our basic results about the complexity of the Forward Chaining construction. In Section 5 we show how our construction can be modified and used to construct stable models of systems possessing certain consistency property. In Section 6 we show how our results of Sections 3 and 4 can be used for Default Logic. In Section 7 we show how our Forward Chaining construction relates to stratification of Apt, Blair and Walker [ABW87]. Finally in Section 8 we indicate how our construction can be coupled with other constructions of models, for instance the constructions of Fitting [Fi85,Fi98] or van Gelder, Ross and Schlipf [VGRS91].

2 Some auxiliary information on logic programs

A *definite logic program* consists of clauses of the form

$$a \leftarrow a_1, \ldots, a_m$$

where a, $a_1, \ldots, a_m$ are atoms of some underlying language. We call such clauses *Horn program clauses* or simply Horn clauses. The set of atoms occurring in some clause of P is called the Herbrand base of P, and is denoted by H_P. We will be dealing here with the propositional case only.

Definition 2.1 A subset $M \subseteq H_P$ is called a model of a set of program clauses P if for all clauses $a \leftarrow a_1, \ldots, a_m$ of P, $a_1, \ldots, a_m \in M$ implies $a \in M$.

A *general logic program* consists of clauses of the form

$$C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_n. \qquad (1)$$

where $a_1, \ldots, a_m$, $b_1, \ldots, b_n$ are atoms of some underlying language. Here $a_1, \ldots, a_n$ are called the *premises* of clause C, $b_1, \ldots, b_m$ are called the *constraints* of clause C, and a is called the *conclusion* of clause C. For any clause C as in (1), we shall write $prem(C) = \{a_1, \ldots, a_n\}$, $cons(C) = \{b_1, \ldots, b_m\}$, and $c(C) = a$. Either $prem(C)$, $cons(C)$, or both may be empty. If $prem(r) = cons(r) = \emptyset$, then the clause r is called an *axiom*.

Each Horn program can be identified with the a general program in which every clause has an empty set of constraints.

Definition 2.2 A subset $M \subseteq H_P$ is called a *model* of P if for all $C = a \leftarrow a_1, \ldots, a_m \neg b_1, \ldots, \neg b_n \in P$, whenever all the premises $a_1, \ldots . a_n$ of C are in M and all the constraints $b_1, \ldots, b_m$ of C are not in M, then the conclusion a of C belongs to M.

For general programs the set of models is not generally closed under arbitrary intersections as in the monotone case. But models are closed under intersections of descending chains. Since H_P is model, by the Kuratowski-Zorn Lemma, there is at least one model minimal among those containing I for any $I \subseteq H_P$.

Given sets $M \subseteq H_P$ and $I \subseteq H_P$, an *M-deduction* of c from I in P is a finite sequence $\langle c_1, \ldots, c_k \rangle$ such that $c_k = c$ and for all $i \leq k$, each c_i either

(i) belongs to I, or
(ii) is the conclusion of an axiom, or
(iii) is the conclusion of a clause $C \in P$ such that all the premises of C are included in $\{c_1, \ldots, c_{i-1}\}$ and all constraints of C are in $H_P - M$ (see [MT93], also [RDB89]).

An *M-consequence* of I is an element of H_P occurring in some M-deduction from I. Let $C_M(I)$ be the set of all M-consequences of I in P. Clearly I is a subset of $C_M(I)$. However note that M enters solely as a restraint on the use of the clauses which may be used in an M-deduction from I. A single constraint in a clause in P may be in M and thus prevent the clause from ever being applied in an M-deduction from I, even though all the premises of that clause occur earlier in a deduction. Thus M contributes no members directly to $C_M(I)$, although members of M may turn up in $C_M(I)$ by an application of a clause which happens to have its conclusion in M. For a fixed M, the operator $C_M(\cdot)$ is monotonic. That is, if $I \subseteq J$, then $C_S(M) \subseteq C_M(J)$. Also, $C_M(C_M(I)) = C_M(I)$. However, for fixed I, the operator $C_M(I)$ is anti-monotonic in the argument M. That is if $M' \subseteq M$, then $C_M(I) \subseteq C_{M'}(I)$.

Generally, $C_M(I)$ is not a model of P. It is perfectly possible that all the premises of a clause be in $C_M(I)$, the constraints of that clause are outside

$C_M(I)$, but a constraint of that clause be in M, preventing the conclusion from being put into $C_M(I)$.

Example 2.1 $H_P = \{a, b, c\}, P = \{a \leftarrow, c \leftarrow a, \neg b\}$, $M = \{b\}$. Then $C_M(\emptyset) = \{a\}$ is not a model of P. □

However, the following holds; see [MNR90].

Proposition 2.3 *If $M \subseteq C_M(I)$, then $C_M(I)$ is model of P.*

We say that $M \subseteq H_P$ is *grounded* in I if $M \subseteq C_M(I)$. We say that $M \subseteq H_P$ is an *stable model of* P over I of I if $C_M(I) = M$. Finally, we say that $M \subseteq H_P$ is a *supported model* of P over I if $C_M(I \cup R) = M$, where R consists of conclusions of those clauses $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_n \in P$ for which $a_1, \ldots, a_n \in M, b_1, \ldots, b_m \notin M$. (Thus we are talking about models of Clark's completion [Cla78], see also [AvE82].)

The notion of groundedness is related to the phenomenon of "reconstruction". M is grounded in I if all elements of M are M-deducible from I (remember that M influences only the negative sides of clauses). M is a stable model of P over I if two things happen. First, every element of M is M-deducible from I, that is, M is grounded in I (this is an analogue of the adequacy property in logical calculi). Second, the converse holds: all the M-consequences of I belong to M (this is the analogue of completeness). Thus stable models are analogues for general programs of the set of all consequences for Horn programs, except that the notion of derivability changes and is self-referring. Both properties (adequacy and completeness) need to be satisfied if we want M to be a stable model.

The third concept, supported model, is a closure property. In the process of constructing $C_M(I)$, M is used only negatively as a restraint. But we can relax our requirements and allow deductions that use M also on the positive side. That is, elements of M are not treated as "axioms", but are used to generate objects from H_P by also testing the positive side of a clause for membership in M. Thus, we get fixpoints of the operator T_P and Clark's completion, see [Apt90]. Gelfond and Lifschitz [GL88] proved that stable models of P are minimal and supported. In particular, stable models of P form an antichain. Moreover, it is easy to see that stable models of P over I are supported models of P over I.

With each clause C of form (1), we associate a Horn clause of form (2)

$$C' = a \leftarrow a_1, \ldots, a_m \tag{2}$$

obtained from C by dropping all the constraints. The clause C' is called the *projection* of clause C. Let M be any subset of H_P and let $G(M, P)$ be the collection of all M-applicable clauses. That is, a clause C belongs to $G(M, P)$ if all the premises of C belong to M and all constraints of C are outside of M. We write $P|_M$ for the collection of all projections of all clauses from $G(M, P)$.

The projection $P|_M$ is a Horn program. Our definition of stable model was different from that given by Gelfond and Lifschitz in [GL88]. It is, however equivalent to it. In particular we have the following.

Theorem 2.4 *$M \subseteq H_P$ is a stable model of P if and only if M is the least model of $P|_M$.*

For the rest of this paper, we shall only consider stable models over $\emptyset$ unless explicitly stated otherwise. We say that M is a *stable model* of P if M is stable model of P over $\emptyset$.

We shall end this section by giving yet another characterization of stable models. For this we need the concept of a proof scheme. A proof scheme for an atom c is a finite sequence

$$p = \langle\langle c_0, C_0, G_0\rangle, \dots, \langle c_m, C_m, G_m\rangle\rangle \tag{3}$$

such that $c_m = c$ and
(1) If $m = 0$ then:
c_0 is a conclusion of a clause

$$C = c_0 \leftarrow \neg b_1, \dots, \neg b_n$$

$C_0 = C$, and $G_0 = cons(C)$.
(This includes the case when c_0 is an axiom that is, when C is of the form $C = c_0 \leftarrow$).
(2) If $m > 0$, then $\langle\langle c_i, r_i, G_i\rangle\rangle_{i=0}^{m-1}$ is a proof scheme of length m and c_m is a conclusion of

$$C = c_m \leftarrow c_{i_0}, \dots c_{i_s}, \neg b_1, \dots, \neg b_r$$

where $i_0, \dots, i_s < m$, $C_m = C$, and $G_m = G_{m-1} \cup cons(C)$.
The atom c_m is called the *conclusion* of p and is written $cln(p)$. The set G_m is called the *support* of p and is written $supp(p)$.

The idea behind this concept is as follows. An M-derivation for P, say p, uses some negative information about M to ensure that the constraints of clauses that were used are outside of M. But this negative information is *finite*, that is, it involves a finite subset of the complement of M. Thus, there exists a finite subset G of the complement of M such that for every set $M_1 \subseteq H_P$, as long as $G \cap M_1 = \emptyset$, p is an M_1-derivation as well. Our notion of proof scheme captures this finitary character of M-derivation.

We can then characterize stable models of P as follows; see [MNR90].

Theorem 2.5 *Let P be a general program. Then M is a stable model of P if and only iff*
(i) for each $c \in M$, there is a proof scheme p such that $cln(p) = c$ and $supp(p) \cap M = \emptyset$ and

(ii) for each $c \notin M$, there is no proof scheme p such that $cln(p) = c$ and $supp(p) \cap M = \emptyset$.

3 The Forward Chaining Construction and Stable Submodels

In this section we shall present our basic Forward Chaining construction which can be applied to any general program P. We shall then establish several basic properties of the Forward Chaining construction.

Given a general program P, we then let $mon(P)$ denote the set of all Horn clauses of P and $nmon(P) = P \setminus mon(P)$. The elements of $nmon(P)$ will be called *nonmonotonic* clauses.

Our Forward Chaining construction will take as an input a program P and a well-ordering $\prec$ of $nmon(P)$. The principal output of the Forward Chaining construction will be a subset $D^{\prec}$ of H_P. Although such subset is not, necessarily, a stable model of P, it will be a stable model of $A^{\prec}$ for a subset $A^{\prec} \subseteq P$. This subset, $A^{\prec}$, will also be computed out of our construction and will be the maximal set of clauses of P for which $D^{\prec}$ is a stable model. We thus call $D^{\prec}$ a *stable submodel* of P.

The first feature of our construction is that in every stage of our construction we will close the sets we construct under $mon(P)$. The point is that stable models are always closed under the operator associated with the Horn part of the program, and the applicability of a clause from $mon(P)$ is not restricted. We shall denote by cl_{mon} the monotone operator of closure under the clauses in $mon(P)$. Thus $cl_{\mathrm{mon}}(I) = T_{mon(P)} \uparrow \omega(I)$ is the least set Z of atoms from H_P such that $I \subseteq Z$ and Z is closed under every clause r of $mon(P)$. That is, if premises of such a clause are all in Z, then its conclusion also belongs to Z. The second important aspect of our construction is that when we inspect the clauses of $nmon(P)$ for a possible application, we look at the possible effect of their application on the applicability of those clauses which were previously applied. Rules that may invalidate applicability of previously used clauses are *not* used.

The execution of this idea requires some book-keeping. Our Forward Chaining construction will define two sequences of subsets of H_P: $\langle D_\xi^{\prec} \rangle_{\xi \le |P|^+}$ and $\langle R_\xi^{\prec} \rangle_{\xi \le |P|^+}$. $D_\xi^{\prec}$ will be the set of *elements derived* by stage ξ. $R_\xi^{\prec}$ will be the set of *elements restrained* by stage ξ. Here and below α^+ is the least cardinal greater than α. Thus, if P is countable, then $|P|^+$ is either finite or the first uncountable ordinal. We shall prove, however, that if $|P|$ is countably infinite, then the construction actually stops *below* the first uncountable ordinal and therefore, for denumerable P, the use of nondenumerable cardinals can be eliminated.

In addition, we shall define two sets of clauses, $I^{\prec}$ (for "inconsistent clauses") and $A^{\prec}$ (for "acceptable" clauses). These sets of clauses will depend on previously defined hierarchies.

3.1 Forward Chaining Construction

We now introduce our Forward Chaining construction. This is done by transfinite induction in the most general case. Note that in case when H_P is finite our construction terminates in finite number of steps. In the infinite case the situation is no different from induction used to in other areas of Computer Science, e.g. Buchberger's construction of Grobner bases, where the algorithms are performed on well-founded ordering of ordinal greater than ω, or Blair's construction of the largest fixpoint for a definite program. We believe that the area of logic programming is no exception. We shall prove below (Proposition 3.9) that if P is countable, then the stable models can be computed with orderings of type $\leq \omega$. Stable submodels, as introduced below, in general, do not share this property.

Definition 3.1 Let P be a general program and let $\prec$ be a well-ordering of *nmon*(P). We define two sequences of sets of atoms from H_P, $\langle D_\xi \rangle$ as well as $\langle R_\xi \rangle$. The set D_ξ is the set of atoms *derived* by stage ξ and R_ξ is the set of atoms *rejected* by the stage ξ.

(i) $D_0^{\prec} = cl_{\text{mon}}(\emptyset)$, $R_0^{\prec} = \emptyset$;
(ii) If $\gamma = \beta + 1$ and there is a clause $C \in nmon(P)$ such that

$$prem(C) \subseteq D_\beta^{\prec}, \quad (\{c(C)\} \cup cons(C)) \cap D_\beta^{\prec} = \emptyset$$

and

$$cl_{\text{mon}}(D_\beta^{\prec} \cup \{c(C)\}) \cap (cons(C) \cup R_\beta^{\prec}) = \emptyset$$

(we call such clause *applicable clause*), then let C_γ be the $\prec$-first applicable clause and set

$$D_\gamma^{\prec} = cl_{\text{mon}}(D_\beta^{\prec} \cup \{c(C_\gamma)\}) \quad R_\gamma^{\prec} = R_\beta^{\prec} \cup cons(C_\gamma).$$

If there is no C such that

$$prem(C) \subseteq D_\beta^{\prec}, \quad (\{c(C)\} \cup cons(C)) \cap D_\beta^{\prec} = \emptyset$$

and

$$cl_{\text{mon}}(D_\beta^{\prec} \cup \{c(C)\}) \cap (cons(C) \cup R_\beta^{\prec}) = \emptyset,$$

then set

$$D_\gamma^{\prec} = D_\beta^{\prec} \quad \text{and} \quad R_\gamma^{\prec} = R_\beta^{\prec}$$

(iii) If γ is a limit ordinal, then

$$D^{\prec}_{\gamma} = \bigcup_{\xi<\gamma} D^{\prec}_{\xi} \quad \text{and} \quad R^{\prec}_{\gamma} = \bigcup_{\xi<\gamma} R^{\prec}_{\xi}.$$

(iv) Finally let

$$D^{\prec} = D^{\prec}_{|P|^+} = \bigcup_{\xi<|P|^+} D^{\prec}_{\xi} \quad \text{and} \quad R^{\prec} = R^{\prec}_{|P|^+} = \bigcup_{\xi<|P|^+} R^{\prec}_{\xi}.$$

Sets $D^{\prec}$ and $R^{\prec}$ are sets of atoms *derived* and *rejected* during the forward chaining construction along the well-ordering $\prec$.

We define the set of inconsistent clauses, $I^{\prec}$, and the set of consistent clauses, $A^{\prec}$, relative to ordering $\prec$ as follows:

5. C is *inconsistent with* $\prec$ (or simply *inconsistent* if $\prec$ is fixed) if $prem(C) \in D^{\prec}$, $(\{c(C)\} \cup cons(C)) \cap D^{\prec} = \emptyset$, but $cl_{\text{mon}}(D^{\prec} \cup \{c(C)\}) \cap (cons(C) \cup R^{\prec}) \neq \emptyset$. $I^{\prec} = \{C \in P : \ C \text{ is inconsistent with } \prec\}$;
6. $A^{\prec} = P \setminus I^{\prec}$

We then say that a subset $D \subseteq H_P$ is a *stable submodel* of P, if there is a well-ordering $\prec$ of $nmon(P)$ such that $D = D^{\prec}$.

The following observations should be clear: First, the clause that is used for construction of $D^{\prec}_{\gamma+1}$ from $D^{\prec}_{\gamma}$ is different from any clause used before in the construction. Therefore, by cardinality argument, the construction, eventually, stabilizes.

Next, both hierarchies $\langle D^{\prec}_{\xi} \rangle$ and $\langle R^{\prec}_{\xi} \rangle$ are increasing. Moreover, it is easy to prove by induction on ξ that $D^{\prec}_{\xi} \cap R^{\prec}_{\xi} = \emptyset$. Therefore $D^{\prec} \cap R^{\prec} = \emptyset$.

The sets $R^{\prec}_{\xi}$ accumulate the restraints of all clauses applied during the construction. Since $D^{\prec} \cap R^{\prec} = \emptyset$, the applicability of clauses applied during the construction is preserved at the end. This immediately implies the following result. First, let $P_{\alpha} = \{C_{\xi} : \xi < \alpha\}$, $P^{*} = \{C_{\alpha} : \alpha < |P|^+ \text{ and } C_{\alpha} \text{ is defined}\}$. We have

Proposition 3.2 *(i) $D^{\prec}_{\xi}$ is a stable model of P_{ξ}*
(ii) $D^{\prec}$ is a stable model of P^{*}.

Proof: For (1), note that it is easy to see that our construction ensures that $cons(C_{\alpha}) \subseteq R_{\xi}$ for all $\alpha \leq \xi$. It then follows that if

$$C_{\alpha} = c \leftarrow a_1, \ldots, a_k \neg b_1, \ldots, \neg b_m,$$

then $\overline{C_{\alpha}} = c \leftarrow a_1, \ldots, a_k$ is a clause in the projection of P_{ξ} relative to D_{ξ}, $P_{\xi} \mid_{D^{\prec}_{\xi}}$. But it is then straightforward to prove that $D^{\prec}_{\xi}$ is the closure of $\emptyset$ in $P_{\xi} \mid_{D^{\prec}_{\xi}}$ and hence $D^{\prec}_{\xi}$ is a stable model of P_{ξ}.

(2) follows from (1) since $D^{\prec} = D^{\prec}_{|P|^+}$ and $P^{\prec} = P^{\prec}_{|P|^+}$. □

We now have a result showing that the set $D^{\prec}$ we produced in the Forward Chaining construction behaves as promised:

Theorem 3.3 *Let P be a general program. Let $\prec$ be a well-ordering of the set $nmon(P)$. Then $D^{\prec}$ is a stable model of $A^{\prec}$. Hence if $I^{\prec} = \emptyset$, then $D^{\prec}$ is a stable model of P.*

Proof: We want to show that $C_{D^{\prec}}(\emptyset) = D^{\prec}$ in the program $A^{\prec}$. This requires two lemmas.

Lemma 3.4 *If $x \in D^{\prec}_{\gamma}$, then there is a sequence $\langle x_1, \ldots, x_n \rangle$ such that $x_n = x$ and for all $i \leq n$, either*

(I) *there is a clause $C = x_i \leftarrow \neg b_1, \ldots, \neg b_n \in A^{\prec}$ such that $\{b_1, \ldots, b_n\} \cap D^{\prec} = \emptyset$ or*
(II) *there is a clause $C = x_i \leftarrow x_{i_1}, \ldots, x_{i_k} \neg b_1, \ldots, \neg b_n \in A^{\prec}$ such that $i_1, \ldots, i_k < i$ and $\{b_1, \ldots, b_n\} \cap D^{\prec} = \emptyset$.*

Proof: We proceed by transfinite induction on γ.

Case 1: $\gamma = 0$. Then $D^{\prec}_0 = cl_{\text{mon}}(\emptyset)$ so that if $x \in D^{\prec}_0$, there is a sequence $\langle x_1, \ldots, x_n \rangle$ with $x_n = x$ such that for all $i \leq n$, either there is a clause $C = x_i \leftarrow \in mon(P)$ or there is a clause $C = x_i \leftarrow x_{i_1}, \ldots, x_{i_k} \in mon(P)$ such that $i_1, \ldots, i_k < i$. Then $\langle x_1, \ldots, x_n \rangle$ is our desired sequence for x.

Case 2: $\gamma = \beta + 1$. Assume the lemma holds for $D^{\prec}_{\beta}$. If $D^{\prec}_{\gamma} = D^{\prec}_{\beta}$, there is nothing to prove. Otherwise, there is a clause

$$C_{\gamma} = c \leftarrow a_1, \ldots, a_p, \neg e_1, \ldots, \neg e_m$$

such that $a_1, \ldots, a_p \in D^{\prec}_{\beta}, e_1, \ldots, e_m \in R^{\prec}_{\gamma}$ and $D^{\prec}_{\gamma} = cl_{\text{mon}}(D^{\prec}_{\beta} \cup \{c\})$. Now suppose $x \in D^{\prec}_{\gamma}$. Then, since $x \in cl_{\text{mon}}(D^{\prec}_{\beta} \cup \{c\})$, there is a sequence $\langle y_1, \ldots, y_n \rangle$ with $y_n = x$ such that either

(a) $y_i \in D^{\prec}_{\beta} \cup \{c\}$,
(b) there is a clause $C = y_i \leftarrow \in mon(P)$, or
(c) there is a clause $C = y_i \leftarrow y_{i_1}, \ldots, y_{i_k} \in mon(P)$ such that $i_1, \ldots, i_k < i$.

Now by induction, if $x \in D^{\prec}_{\beta}$, there is a sequence $\langle z^x_1, \ldots, z^x_{t_x} \rangle$ such that for all $j \leq t_x$ either

(i) there is a clause $C = z^x_j \leftarrow \neg b_1, \ldots, \neg b_s \in A^{\prec}$ such that $\{b_1, \ldots, b_s\} \cap D^{\prec} = \emptyset$, or
(ii) there is a clause $C = z_i \leftarrow z_{i_1}, \ldots, z_{i_k} \neg b_1, \ldots, \neg b_s \in A^{\prec}$ such that $\{b_1, \ldots, b_s\} \cap D^{\prec} = \emptyset$ and $i_1, \ldots, i_k < j$.

Now consider the sequence $\langle y_1, \ldots, y_n \rangle$. For each y_i, we shall define a sequence w_{y_i} as follows: First suppose y_i satisfies Case (a). Then if $y_i \in D^{\prec}_{\beta}$, we let $w_{y_i} = \langle z^{y_i}_1, \ldots, z^{y_i}_{t_{y_i}} \rangle$ as described above. If $y_i = c$, then we let $w_{y_i} = w_c = \langle z^{a_1}_1, \ldots, z^{a_1}_{t_{a_1}}, \ldots, z^{a_p}_1, \ldots, z^{a_p}_{t_{a_p}}, c \rangle$ where for each $j \leq n$, $\langle z^{a_j}_1, \ldots, z^{a_j}_{t_{a_j}} \rangle$ is the sequence satisfying **(i)** and **(ii)** for $x = a_j$. Such a sequence $\langle z_1^{a_j}, \dots, z_{t_{a_j}}^{a_j} \rangle$ exists because by the definition of C_γ, $a_j \in D_\beta^\prec$. We claim that the entire sequence w_c satisfies conditions **(I)** and **(II)**. Certainly each of the subsequences $\langle z_1^{a_j}, \dots, z_{ta_j}^{a_j} \rangle$ satisfy **(I)** and **(II)**. Now

$$C_\gamma = c \leftarrow a_1, \dots, a_p, \neg e_1, \dots, \neg e_m \in A^\prec$$

and $\{e_1, \dots, e_m\} \subset R_\gamma^\prec \subseteq R^\prec$ by construction. Thus $\{e_1, \dots, e_m\} \cap D^\prec = \emptyset$ since $R^\prec \cap D^\prec = \emptyset$. Thus C_γ shows that c satisfies condition **(II)** for w_c.

Finally if y_i satisfies Case 2(b) or 2(c), then we let $w_{y_i} = \langle y_i \rangle$. It then follows that if w is the concatenation of the sequences $w_{y_1}, \dots, w_{y_n}$, then w satisfies conditions **(I)** and **(II)** for x.

Case 3: γ is a limit ordinal. Then if $x \in D_\gamma^\prec = \bigcup_{\beta \prec \gamma} D_\beta^\prec$, then for some $\beta < \gamma, x \in D_\beta^\prec$. Thus by induction, there is a sequence $\langle x_1, \dots, x_n \rangle$ satisfying **(I)** and **(II)** for x. □

Note that Lemma 3.4 implies that $D^\prec \subseteq C_{D^\prec}(\emptyset)$ relative to $A^\prec$ To prove that $C_{D^\prec}(\emptyset) \subseteq D^\prec$, we need only prove the following.

Lemma 3.5 *Suppose $\langle x_1, \dots, x_n \rangle$ is a sequence such that for all $i \leq n$, either*

(I) there is a clause $C_i = x_i \leftarrow \neg b_1, \dots, \neg b_s \in A^\prec$ such that $\{b_1, \dots, b_s\} \cap D^\prec = \emptyset$ or

(II) there is a clause $C_i = x_i \leftarrow x_{i_1}, \dots, x_{i_k} \neg b_1, \dots, \neg b_s \in A^\prec$ such that

$$\{b_1, \dots, b_s\} \cap D^\prec = \emptyset \quad \textit{and } i_1, \dots, i_k < i.$$

Then $x_n \in D^\prec$.

Proof: We proceed by induction on n. We can thus assume that $x_1, \dots, x_{n-1} \in D^\prec$. Hence there is some stage γ_0 such that $x_1, \dots, x_{n-1} \in D_{\gamma_0}^\prec$. Now suppose, for a contradiction, that $x_n \notin D^\prec$. Note that at successor stages $\gamma = \beta + 1$ of our construction, if r_γ is defined, then

$$r_\gamma = c \leftarrow a_1, \dots, a_n, \neg b_1, \dots, \neg b_m$$

satisfies $c \notin D_\beta^\prec$ and $c \in D_\gamma^\prec$. It follows that if $\gamma_1 \neq \gamma_2$ are two successor ordinals smaller than $|nmon(P)|^+$ such that r_{γ_1} and r_{γ_2} are defined, then $r_{\gamma_1} \neq r_{\gamma_2}$. Now by a cardinality argument, there must be a stage $\alpha > \gamma_0$ such that $\alpha = \beta' + 1$ is a successor ordinal and all clauses C such that $C \prec r_n$ do not satisfy the criteria to be r_α. Thus at stage $\alpha, r_n = x_n \leftarrow x_{i_1}, \dots, x_{i_k}, \neg b_1, \dots, \neg b_s$ satisfies that $\{x_{i_1}, \dots, x_{i_k}\} \subseteq D_\beta^\prec, \{x_n, b_1, \dots, b_s\} \cap D_{\beta'}^\prec = \emptyset$. Then the only way that we would not pick $r_\alpha = r_n$ is either if $x_n \in D_{\beta'}^\prec$, in which case we are done, or if $cl_{\text{mon}}(D_{\beta'}^\prec \cup \{x_n\}) \cap (\{b_1, \dots, b_s\} \cup R_{\beta'}^\prec) \neq \emptyset$. But then clearly $cl_{\text{mon}}(D^\prec \cup \{x_n\}) \cap (\{b_1, \dots, b_s\} \cup R^\prec) \neq \emptyset$ which would mean $r_n \in I^\prec$. But by assumption $r_n \in A^\prec = P - I^\prec$. Thus we must conclude that $r_\alpha = r_n$ and hence $x_n \in D^{\prec}_{\alpha} \subseteq D^{\prec}$ which contradicts our assumption that $x_n \notin D^{\prec}$. Hence $x_n \in D^{\prec}$ as desired. $\square$

Lemmas 3.4 and 3.5 imply the Proposition 3.3.

Note that our original definition of $A^{\prec}$ depends on both $D^{\prec}$ and $R^{\prec}$. This is because $I^{\prec}$ depends on both $D^{\prec}$ and $R^{\prec}$. Since, however, $D^{\prec}$ is a stable model of $A^{\prec}$, it follows that whenever $C \in A^{\prec}$, then $D^{\prec}$ is closed under C. That is, if $prem(C) \subseteq D^{\prec}$, then either $cons(C) \cap D^{\prec} \neq \emptyset$ or $c(C) \in D^{\prec}$. On the other hand, whenever $C \in I^{\prec}$ (that is $C \notin A^{\prec}$), then $c(C) \notin D^{\prec}$, $prem(C) \subseteq D^{\prec}$ but $cl_{\text{mon}}(D^{\prec} \cup \{c(C)\}) \cap cons(C) \neq \emptyset$ or $cl_{\text{mon}}(D^{\prec} \cup \{c(C)\}) \cap R^{\prec} \neq \emptyset$. In this latter case $cl_{\text{mon}}(D^{\prec} \cup \{c(C)\}) \cap (H_P \setminus D^{\prec}) \neq \emptyset$. Therefore we get the following characterization of $A^{\prec}$ which depends only on $D^{\prec}$.

Corollary 3.6 *If P is a general program and $\prec$ is a well-ordering of of the set $nmon(P)$, then*

$$A^{\prec} = \{C \in P : prem(C) \not\subseteq D^{\prec} \ \ or \ \ cl_{mon}(D^{\prec} \cup \{c(C)\}) \cap (H_P \setminus D^{\prec}) \neq \emptyset\}$$

In Section 5, we shall describe a set of programs P which we call *FC-normal* programs which have the property that when the Forward Chaining construction is applied to P, $I^{\prec}$ is always empty. FC-normal programs were introduced in [MNR93b]. FC-normal programs are guaranteed to have stable models. These systems are generalizations of Reiter's normal default logic [Rei80]. The properties of FC-normal program are proved in [MNR93b] so that in section 5, we shall simply give basic definitions and results of [MNR93b].

We define the set of nonmonotonic generating clauses for a set $M \subseteq H_P$, $NG(M, P)$.

Definition 3.7 Let P be a general program. Let $M \subseteq H_P$.

$$NG(M,P) = \{C \in nmon(P) : prem(C) \subseteq M, cons(C) \cap M = \emptyset\}$$

Thus $NG(M, P) = G(M, P) \cap nmon(P)$.

Next we show the completeness of our construction, that is that every stable model is a stable submodel.

Theorem 3.8 *If P is a general program, then every stable model of P is a stable submodel of P. That is, if M is a stable model of P, then there exists a well-ordering $\prec$ of $nmon(P)$ such that $D^{\prec} = M$. In fact, for every well-ordering $\prec$ such that $NG(M, P)$ forms an initial segment of $\prec$, $D^{\prec} = M$.*

Proof: First of all notice that since M is a stable model of P, M is a supported model of P. Thus it is generated from the conclusions of nonmonotonic generating clauses by means of monotonic clauses.

Next, let $\prec$ be a well-ordering of $nmon(P)$ such that $NG(M, P)$ forms an initial segment of $\prec$. Let δ be the order type of $\prec$. Then we can write $nmon(P)$ as $nmon(P) = \{n_\alpha : \alpha < \delta\}$. By our assumption there is a $\gamma \leq \delta$ such that $NG(M,P) = \{n_\alpha : \alpha < \gamma\}$. Let $\mu = \gamma^+$ and let $\{(D_\alpha^\prec, R_\alpha^\prec) : \alpha \in \mu\}$ be the sequence constructed by the Forward Chaining construction for P relative to the well-ordering $\prec$. Then we claim that $D_\mu^\prec = M$.

First it is easy to show by induction that $cl_{\text{mon}}(D_\alpha^\prec) = D_\alpha^\prec$ for all α. Next we claim that if $\alpha < \mu$, then $D_\alpha^\prec \subseteq M$ and moreover if $D_\alpha^\prec \neq D_{\alpha+1}^\prec$, then $r_{\alpha+1} = n_\theta$ for some $\theta < \gamma$, i.e., $r_{\alpha+1} \in NG(M,P)$. That is, suppose by induction that $D_\beta^\prec \subseteq M$ for all $\beta < \alpha$. Then if α is a limit ordinal, $D_\alpha^\prec = \bigcup_{\beta<\alpha} D_\beta^\prec \subseteq M$. If α is a successor ordinal, we can assume by induction that $\alpha = \eta + 1$ where $D_\eta^\prec \subseteq M$ and that $r_\beta \in NG(M,P)$ for all $\beta < \alpha$ such that r_β is defined. Since $r_\beta \in NG(M,P)$, we know that $cons(r_\beta) \cap M = \emptyset$. But $R_\eta^\prec = \bigcup_{\beta \leq \eta} cons(r_\beta)$ so that $R_\eta^\prec \cap M = \emptyset$ as well. Now consider $D_\eta^\prec$. If $D_\eta^\prec = M$, then for any clause $r = c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, b_m$ either $\{a_1, \ldots, a_n\} \not\subseteq M$ or $\{c, b_1, \ldots, b_m\} \cap M \neq \emptyset$ since M is a stable model. But this means that $r_{\eta+1}$ must be undefined so that $D_\alpha^\prec = D_\eta^\prec = M$. If $D_\eta^\prec \subset M$, then consider some $a \in M \setminus D_\eta^\prec$. Since $a \in M$, there is some minimal proof scheme $p = \langle\langle a_0, \bar{r}_0, G_0\rangle, \ldots, \langle a_m, \bar{r}_m, G_m\rangle\rangle$ where $a_m = a$ and $G_m \cap M = \emptyset$ which witnesses that $a \in M$. Since $a \notin D_\eta^\prec$, there must be some $k \leq m$ such that $a_0, \ldots, a_{k-1} \in D_\eta^\prec$ and $a_k \notin D_\eta^\prec$. Then consider $\bar{r}_k = a_k \leftarrow a_{i_0}, \ldots, a_{i_j}, \neg b_1, \ldots, \neg b_t$ where $i_0, \ldots, i_j < k$.

Now it cannot be that $\{b_1, \ldots, b_t\} = \emptyset$ since otherwise $a_k \in cl_{\text{mon}}(D_\eta^\prec) = D_\eta^\prec$. Thus $\{b_1, \ldots, b_t\} \neq \emptyset$. But since $\{b_1, \ldots, b_t\} \subseteq G_m$ and $G_m \cap M = \emptyset$, it must be the case that $\{b_1, \ldots, b_t\} \cap M = \emptyset$ so that certainly $\{b_1, \ldots, b_t\} \cap D_\eta^\prec = \emptyset$. Moreover, $D_\eta^\prec \cup \{a_k\} \subseteq M$ so that $cl_{\text{mon}}(D_\eta^\prec \cup \{a_k\}) \subseteq M$. Also $(\{b_1, \ldots, b_t\} \cup R_\eta^\prec) \cap M = \emptyset$ so that $cl_{\text{mon}}(D_n^\prec \cup \{a_k\}) \cap (\{b_1, \ldots, b_t\} \cup R_n^\prec) = \emptyset$. Thus $\bar{r}_k$ is a candidate to be r_α at stage α. But $\bar{r}_k \in NG(M,P)$ so that $\bar{r}_k = n_\theta$ for some $\theta < \gamma$. Hence by construction $r_\alpha = n_\lambda$ for some $\lambda \leq \theta < \gamma$. Thus $r_\alpha \in NG(M.P)$ and $cln(r_\alpha) \in M$. But this means $D_\eta^\prec \cup \{cln(r_\alpha)\} \subseteq M$ so that $D_\alpha^\prec = cl_{\text{mon}}(D_\eta^\prec \cup \{cln(r_\alpha)\}) \subseteq M$.

It follows that $D_\mu^\prec \subseteq M$. That is, μ is a limit ordinal so that $D_\mu^\prec = \bigcup_{\alpha\in\mu} D_\alpha^\prec$ and we have proved $D_\alpha^\prec \subseteq M$ for all $\alpha \in \mu$. We claim that it must be the case that $D_\mu^\prec = M$ for otherwise $D_\mu^\prec \subset M$ and hence for all $\alpha \in \mu, D_\alpha^\prec \subset M$. But our argument above shows that if $D_\alpha^\prec \subset M$, then $D_\alpha^\prec \subset D_{\alpha+1}^\prec$, and $r_{\alpha+1} \in NG(M,P)$. This fact, in turn, will allow us to prove by induction on the length of a minimal proof scheme that for all $r \in NG(M,P), cln(r) \in D_\mu^\prec$. That is, suppose $p = cln(r)$ for some $r \in NG(M,P)$. Now since $p \in M$, there is a minimal proof scheme $p = \langle\langle a_0, \bar{r}_0, G_0\rangle, \ldots, \langle a_m, \bar{r}_m, G_m\rangle\rangle$ where $G_m \cap M = \emptyset$ and $a_m = p$. Now we can assume by induction that if c is the conclusion of a minimal proof scheme q such that $supp(q) \cap M = \emptyset$ and the length of $q < m$, then $c \in D_\mu^\prec$. Thus we can assume $a_0, \ldots, a_{m-1}$ are in $D_\mu^\prec$. Hence there is some $\alpha < \mu$ such that $\{a_0, \ldots, a_{m-1}\} \subseteq D_\alpha^\prec$. Then consider $\bar{r}_m = p \leftarrow a_{i_1}, \ldots, a_{i_k}, \neg b_1, \ldots, \neg b_t$ where $i_j, \ldots, i_k < m$. Since $\{a_{i_1}, \ldots, a_{i_k}\} \subseteq D_\alpha^\prec$

and $\{b_1, \ldots, b_t\} \cap M \neq \emptyset$, it will follow that for all β such that $p \notin D^{\prec}_{\beta}$ and $\alpha \leq \beta < \mu$, $\bar{r}_m$ is a candidate to be $r_{\beta-1}$ at stage β. Now $\bar{r}_m = n_\lambda$ for some $\lambda < \gamma$. Thus at such at stage β, it must be the case that either $r_{\beta+1} = n_\lambda = \bar{r}_m$ or $r_{\beta+1} = n_\theta$ for some $\theta < \lambda$. Moreover if $\beta_1 \neq \beta_2$ and $\bar{r}_{\beta_1}$ and $\bar{r}_{\beta_2}$ are defined, then we have previously observed that $r_{\beta_1} \neq r_{\beta_2}$. If follows that because μ is cardinal and hence $card(\{n_\theta : \theta \leq \lambda\}) < card(\mu)$, there must be a stage β such that $\alpha \leq \beta < \mu$ and either $p \in D^{\prec}_{\beta}$ or $\bar{r}_m = r_{\beta+1}$ In either case, p will be in $D^{\prec}_{\beta+1} \subseteq D^{\prec}_{\mu}$. It follows that $\{cln(r) : r \in NG(M, P)\} \subseteq D^{\prec}_{\mu} \subseteq M$. But then $M = cl_{\text{mon}}(\{cln(r) : r \in NG(M, P)\}) \subseteq cl_{\text{mon}}(D^{\prec}_{\mu}) = D^{\prec}_{\mu} \subseteq M$. Thus $D^{\prec}_{\mu} = M$ as claimed.

Now if $\mu = |P|^+$, then $D^{\prec}_{\mu} = D^{\prec} = M$. If $\mu < |P|^+$, then we know that $D_\mu = M$. But by our observation above, whenever $D_\alpha = M$, $r_{\alpha+1}$ is undefined. Thus $r_{\mu+1}$ is undefined and hence $D^{\prec}_{\alpha} = D^{\prec}_{\mu} = M$ for all $\mu \leq \alpha \leq |P|^+$. Thus in either case $D^{\prec} = M$. □

Next, we shall consider some examples.

Example 3.1 Let $H_P = \{a, b, c, d, e, f\}$ and let P consist of the following clauses:

(i) $a \leftarrow$
(ii) $b \leftarrow c$
(iii) $c \leftarrow a, \neg d$
(iv) $d \leftarrow b, \neg c$
(v) $e \leftarrow c, \neg f$
(vi) $f \leftarrow c, \neg e$

Here, $mon(P)$ consists of clauses (1) and (2), whereas $nmon(P)$ consists of clauses (3), (4), (5), and (6).
Let $\prec$ of $nmon(P)$ be defined as (3) $\prec$ (4) $\prec$ (5) $\prec$ (6). Then the construction of sets $D^{\prec}_{n}$ and $R^{\prec}_{n}$ is as follows:
Stage 0 $D^{\prec}_{0} = cl_{\text{mon}}(\emptyset) = \{a\}$, $R^{\prec}_{0} = \emptyset$.
Stage 1 $r_1 = (3)$, $D^{\prec}_{1} = cl_{\text{mon}}(\{a\} \cup \{c\}) = \{a, b, c\}$, $R^{\prec}_{1} = \{d\}$.
Stage 2 $r_2 = (5)$, $D^{\prec}_{2} = \{a, b, c, e\}$, and $R^{\prec}_{2} = \{d, f\}$.
Stage 3 At this stage our construction stabilizes.
It is easy to see that $I^{\prec} = \emptyset$ so $D^{\prec} = D^{\prec}_{2}$ is an stable model of P.

Now, let $\prec'$ be an ordering of $nmon(P)$ as follows: (4) $\prec'$ (3) $\prec'$ (6) $\prec'$ (5). Here, the construction of $D^{\prec'}$ produces these stages:
Stage 0 $D^{\prec'}_{0} = cl_{\text{mon}}(\emptyset) = \{a\}$, $R^{\prec'}_{0} = \emptyset$.
Stage 1 $r_1 = (3)$, $D^{\prec'}_{1} = cl_{\text{mon}}(\{a\} \cup \{c\}) = \{a, b, c\}$, $R^{\prec'}_{1} = \{d\}$.
Stage 2 $r_2 = (6)$, $D^{\prec'}_{2} = \{a, b, c, f\}$, and $R^{\prec'}_{2} = \{d, e\}$.
Stage 3 At this stage our construction stabilizes.
Again, it is easy to see that $I^{\prec'} = \emptyset$ so $D^{\prec'} = D^{\prec'}_{2}$ is a stable model of P. These are the only stable models of P and the construction with any well-ordering will produce one of these. □

Let us look at another example.

Example 3.2 Let $H_P = \{a.b.c.d,e\}$ and let P consist of the following clauses:

(i) $a \leftarrow$
(ii) $e \leftarrow b$
(iii) $c \leftarrow d$
(iv) $d \leftarrow a, \neg b$
(v) $b \leftarrow a, \neg d$
(vi) $d \leftarrow e, \neg c$
(vii) $b \leftarrow c, \neg e$

Here, $mon(P)$ consists of clauses (1), (2) and (3), whereas $nmon(P)$ consists of clauses (4), (5), (6), and (7).
Let $\prec$ of $nmon(P)$ be defined as (4) $\prec$ (5) $\prec$ (6) $\prec$ (7). Then the construction of sets $D_n^{\prec}$ and $R_n^{\prec}$ is as follows:
Stage 0 $D_0^{\prec} = cl_{\text{mon}}(\emptyset) = \{a\}$. $R_0^{\prec} = \emptyset$.
Stage 1 $r_1 = (4)$, $D_1^{\prec} = cl_{\text{mon}}(\{a\} \cup \{d\}) = \{a, c, d\}$. $R_1^{\prec} = \{b\}$.
Stage 2 r_2 is undefined and the construction stops at this stage. Hence $D^{\prec} = D_1^{\prec} = \{a, c, d\}$. We check that $I^{\prec}$ consists of the clause (7). $A^{\prec}$ consists of clauses (1)-(6). Thus $\{a, c, d\}$ is a stable model of $(1)-(6)$ but not of $(1)-(7)$.

Now, let $\prec'$ be an ordering of $nmon(P)$ as follows: (7) $\prec'$ (6) $\prec'$ (5) $\prec'$ (4). Here, the construction of $D^{\prec'}$ produces these stages:
Stage 0 $D_0^{\prec'} = cl_{\text{mon}}(\emptyset) = \{a\}$. $R_0^{\prec'} = \emptyset$.
Stage 1 $r_1 = (5)$, $D_1^{\prec'} = cl_{\text{mon}}(\{a\} \cup \{b\}) = \{a, b. e\}$, $R_1^{\prec} = \{d\}$.
Stage 2 r_2 is undefined and the construction stops at this stage. Hence $D^{\prec'} = D_1^{\prec'} = \{a, b, e\}$. We check that $I^{\prec'}$ consists of the clause (6). $A^{\prec'}$ consists of clauses (1)-(5) and (7). Thus $\{a. c. d\}$ is a stable model of $(1)-(5), (7)$ but not of $(1)-(7)$. In fact, it is easy to check that in any well-ordering of $nmon(P)$, either clause (4) or clause (5) will be $r_1$and hence $\{a, b, d\}$ and $\{a, b, e\}$ are the only two stable submodels of P. Thus P has no stable models. This example shows that while P is inconsistent, P has two maximal subprograms which are consistent. □

While we stated Theorem 3.3 and Theorem 3.8 in full generality, we are most interested in the case when program P is finite or countable. In this case we can show that to construct stable models via forward chaining, one need consider orderings of type smaller or equal of order type ω. So let us assume that $|P| \leq \omega$. Note that it follows easily from Propositions 3.8 that it is enough to consider orderings of type $\leq \omega + \omega$. For in this case $NG(M, P)$ can be ordered in type $\leq \omega$ and similarly $nmon(P) \setminus NG(M, P)$ can be ordered in type $\leq \omega$. But it turns out that, actually, only the orderings of type $\leq \omega$ are needed.

Proposition 3.9 *Let P be a program such that $|H_P| \leq \omega$ and let M be a stable model of P. There exists a well-ordering $\prec'$ of nmon(P) in type $\leq \omega$*

such that $D^{\prec'} = M$. Moreover the forward Chaining construction stabilizes in at most ω steps.

Proof: Consider $NG(M,P)$. If $NG(M,P)$ is finite, then let $\prec$ be an ordering of $nmon(P)$ of order type ω such that the set of clauses in $NG(M,P)$ form an initial segment under $\prec$. Such an ordering $\prec$ must exist since $nmon(P)$ is countable and $NG(M,P)$ is finite. Then our proof in Proposition 3.8 shows that in the Forward Chaining construction with respect to $\prec, D_\omega^\prec = M = D^\prec$. Now if $NG(M,P)$ is infinite, let $n_0, n_1, \ldots$ be a list of $NG(M,P)$ and $d_0, d_1, \ldots$ be a list of $nmon(P) - NG(M,P)$. Let $\prec_1$ be the well-ordering of $nmon(P)$ defined by $n_0 \prec_1 n_1 \prec_1 \ldots \prec_1 d_0 \prec_1 d_1 \prec \ldots$. Again our proof of Proposition 3.8 shows that $D_\omega^{\prec_1} = \bigcup_{n \in w} D_n^{\prec_1} = M$. Since M is a stable model, it follows that for each clause $d_n = c_n \leftarrow a_1^n, \ldots, a_{p_n}^n, \neg b_1^n, \ldots, \neg b_{t_n}^n$ either $\{a_1^n, \ldots, a_{p_n}^n\} \not\subseteq M$ or $\{b_1^n, \ldots, b_{t_n}^n\} \cap M \neq \emptyset$ by our definition of $NG(M,P)$. Then define $\Gamma(d_n) = n$ if $\{a_1^n, \ldots, a_{p_n}^n\} \not\subseteq M$ and $\Gamma(d_n)$ to be the least $m \geq n$ such that $\{b_1^n, \ldots, b_{t_n}^n\} \cap D_m^{\prec_1} \neq \emptyset$ otherwise. Clearly $\Gamma(d_n)$ is defined for all n. Next let $\vartheta(n) = 1 + \max\{k : n_k = r_j \text{ for some } j \leq n\}$ where r_j is the clause defined at stage j of our Forward Chaining construction with respect to $\prec_1$. Finally for each $n \geq 0$, define the rank of d_n, $rank(d_n)$, by $rank(d_n) = \vartheta(\Gamma(d_n))$. The significance of the rank of d_n is that we can insert d_n into the list $n_0, n_1, \ldots$, at any point after $n_{rank(d_n)}$ and remove it from the list $d_0, d_1, \ldots$, to form a new ordering $\prec_2$. Then if we run the Forward Chaining construction with respect to $\prec_2$, the first $\Gamma(d_n)$ steps of the Forward Chaining construction with respect to $\prec_2$ will be identical with the first $\Gamma(d_n)$ steps of the Forward Chaining construction with respect to $\prec_1$. That is, our definition of ϑ ensures that the first $\Gamma(d_n)$ steps of the Forward Chaining construction with respect to $\prec_1$ depends only on clauses $n_0, \ldots, n_{\vartheta(\Gamma(d_n))}$. Thus since $n_0, \ldots, n_{\vartheta(\Gamma(d_n))}$ also are the first $1 + \vartheta(\Gamma(d_n))$ clauses with respect to $\prec_2$, the first $\Gamma(d_n)$ steps of the Forward Chaining construction with respect to $\prec_2$ will stay the same as in the Forward Chaining construction with respect to $\prec_1$.

Hence $D_{\Gamma(d_n)}^{\prec_2} = D_{\Gamma(d_n)}^{\prec_1}$. But our choice $\Gamma(d_n)$ ensures that either

$$\{b_1^n, \ldots, b_{t_n}^n\} \cap D_{\Gamma(d_n)}^{\prec_1} \neq \emptyset$$

or

$$\{a_1^n, \ldots, a_{p_n}^n\} \not\subseteq M.$$

It follows that the insertion of d_n into the list $n_0, n_1, \ldots$ does not effect any step of the Forward Chaining construction up to stage ω. Thus we can conclude $D_\omega^{\prec_1} = D_\omega^{\prec_2} = M$. But then since M is a stable model, $r_{\omega+1}$ must be undefined in both the Forward Chaining constructions with respect to $\prec_1$, and $\prec_2$ and hence $D^{\prec_2} = D^{\prec_1} = M$. Note that our definition of $rank(d_n)$ ensures that for any given rank k, there are only finitely many clauses d_n such that $rank(r_n) = k$. Now construct a new list of the elements of $nmon(P), a_0, a_1, \ldots$, by starting with the list $n_0, n_1, \ldots$ and then inserting all the clauses d_n of rank k, say in increasing order with respect to $\prec_1$, between n_k and n_{k+1}. Then if $\prec$ is the well-ordering of $nmon(P)$ defined by $a_0 \prec a_1 \prec \dots$ it follows by our arguments above that in the Forward Chaining construction with respect to $\prec$, $D_n^{\prec} = D_n^{\prec_1}$ for all $n \in \omega$. Thus $D^{\prec} = D_\omega^{\prec} = M$ as desired. □

We note that Theorem 3.9 does not hold for all stable submodels. That is, the sets $D^{\prec}$ which are *not* stable models may have the property that they can only be obtained by means of orderings of the length $> \omega$. Moreover, in opposition to stable models, stable submodels do not form an antichain.

Example 3.3 Let $H_P = \{a_i : i < \omega\} \cup \{b_i : i < \omega\} \cup \{c_i : i < \omega\} \cup \{p\}$. Let P consist of the following clauses:

(i) $q_i = a_i \leftarrow a_1, \dots, a_{i-1}, \neg b_i$ for $i \geq 1$ (thus $q_1 = a_1 \leftarrow \neg b_1$);
(ii) $s_i = b_i \leftarrow a_i, \neg c_i$ for $i \geq 1$;
(iii) $q_0 = a_0 \leftarrow \neg b_0$;
(iv) $t_i = a_i \leftarrow a_0$ for $i \geq 1$;
(v) $u = p \leftarrow \neg p$.

Clearly, P has no stable model. Now, order the clauses in $nmon(P)$ as follows:

$$q_1 \prec q_2 \prec \dots q_0 \prec s_1 \prec s_2 \dots \prec u$$

It is easy to see that $D^{\prec} = \{a_i : i \in \omega\}$, $R^{\prec} = \{b_i : i \in \omega\}$, and $A^{\prec} = \{q_0, q_1, t_1, q_2, t_2, \dots\}$. However we claim that if $\prec_1$ is an ordering of P of order type ω, then $D^{\prec_1}$ must contain at least one b_i. This implies that $\{a_i : i \in \omega\}$ is not equal to $D^{\prec_1}$ where $\prec_1$ is of order type ω. That is, suppose that $n_0 \prec_1 n_1 \prec_1 n_2 \prec_1 \dots$ is an ordering of P of order type ω. Not that since b_0 is not the conclusion of any clause in P, at any stage k, q_0 is a candidate to become r_k. Since $n_l = q_0$ for some l, it follows that for some $k \leq l+1$, $r_k = q_0$ and hence

$$D_k^{\prec_1} = cl_{\text{mon}}(D_{k-1}^{\prec_1} \cup \{a_0\}) \supseteq \{a_i : i \in \omega\}$$

Since $R_k^{\prec_1}$ is finite, infinitely many of the clauses s_i are applicable at stage $k+1$. However none of the clauses q_i are applicable at stage $k+1$ since $a_i \in D_k^{\prec_1}$ for all $i \in \omega$. Thus $r_{k+1} = s_j$ for some j and hence $b_j \in D_{k+1}^{\prec_1} \subseteq D^{\prec_1}$. (In fact, it is easy to see that all but finitely many b_j will be in $D^{\prec_1}$.) □

Our construction of the set $D^{\prec}$ persists with respect to prolongation of the well-ordering (providing the Horn part is the same).

Proposition 3.10 *Let $P \subset P'$ be two sets of clauses such that $mon(P) = mon(P')$. Let $\prec'$ be a well-ordering of $nmon(P')$ and let $nmon(P)$ be an initial segment in $\prec'$. Finally, let $\prec = \prec' |_P$. Then $D^{\prec} \subseteq D^{\prec'}$ and $R^{\prec} \subseteq R^{\prec'}$.*

Proof: Let α be the least ordinal η such that $D_\eta^{\prec} = D_{\eta+1}^{\prec}$. Let α' be the least ordinal η such that $D_\eta^{\prec'} = D_{\eta+1}^{\prec'}$. It is straightforward to prove by induction on

$\xi < \alpha$ that $r^{\prec}_{\xi} = r^{\prec'}_{\xi}$ and, consequently, $D^{\prec}_{\xi} = D^{\prec'}_{\xi}$ and $R^{\prec}_{\xi} = R^{\prec'}_{\xi}$. This implies that $D^{\prec} = \bigcup_{\xi \leq \alpha} D^{\prec}_{\xi} = \bigcup_{\xi \leq \alpha} D^{\prec'}_{\xi}$ and so $\alpha \leq \alpha'$. Thus $D^{\prec} \subseteq \bigcup_{\xi \leq \alpha'} D^{\prec'}_{\xi} = D^{\prec'}$.

Similar argument shows that $R^{\prec} \subseteq R^{\prec'}$. □

Our argument shows that if one ordering is a prolongation of another, then all the clauses applied in the construction along the smaller well-ordering have also been used in the construction along the longer well-ordering. But what happens with the remaining clauses, those in $nmon(P) \setminus NG(P, D^{\prec})$? These consist of two types of clauses: inconsistent clauses and clauses unused in the construction. Inconsistent clauses may become unused. This will happen, for instance, if their conclusion is also a conclusion of another clause used later in the construction. The clauses unused in the construction along $\prec$ may become inconsistent or remain unused.

Example 3.4 Let P consist of $r_0 = p \leftarrow$, $r_1 = t \leftarrow p, \neg q$ and $r_2 = q \leftarrow s, \neg r$. Let $r_1 \prec r_2$. It is easy to check that $D^{\prec} = \{p, t\}$, $R^{\prec} = \{q\}$, and $I^{\prec} = \emptyset$. Next, let $P' = P \cup \{r_3\}$ where $r_3 = s \leftarrow \neg w$. Let $r_1 \prec' r_2 \prec' r_3$.

Again it is easy to check that $D^{\prec'} = \{p, t, s\}$ and $R^{\prec'} = \{q, w\}$ so that now $r_2 \in I^{\prec'}$. □

Example 3.5 Let P consists of a clause $r = p \leftarrow \neg p$ only. Then with the trivial well-ordering of P the clause r is inconsistent. But when we add a clause $r' = p \leftarrow \neg q$, then, in the construction along any ordering $\prec$, r becomes unused. □

In our presentation we treated our Horn clauses in "absolute fashion". In particular we always took the closure cl_{mon} under *all* the clauses in $mon(P)$. In every step of the Forward Chaining construction we took care of making sure that every level is closed under all the Horn clauses. This does not have to be the case. That is, in a specific application we can treat some Horn clauses as "nonmonotonic clauses without restraints". Specifically, let $R \subseteq mon(P)$. Then we can associate with R a monotonic operator $cl_R(\cdot)$ of closure under the clauses in R. Since $R \subseteq mon(P)$, all the nonmonotonic clauses belong to $P \setminus R$. Let $\prec$ be a well-ordering of $P \setminus R$. We can now execute the Forward Chaining construction with respect to $\prec$. If the set of clauses $I^{\prec}$ is empty then, as before, we get a stable model of P. When $I^{\prec}$ is nonempty it may contain monotonic clauses in $mon(P) \setminus R$.

Example 3.6 Let $H_P = \{a, b, c, d, e\}$. Let $P = \{r_1, r_2, r_3, r_4, r_5\}$ where $r_1 = b \leftarrow a, r_2 = a \leftarrow$, $r_3 = c \leftarrow b, r_4 = d \leftarrow \neg c, r_5 = f \leftarrow d \neg f$.

Here, $mon(P) = \{r_1, r_2, r_3\}$. $cl_{\text{mon}}(\emptyset) = \{a, b, c\}$, the clauses r_4, r_5 are unused under any well-ordering and the only set computed here is $\{a, b, c\}$ which is an stable model of P. But with $R = \{r_1, r_2\}$ and an ordering $\prec$ in which the clause r_3 follows the remaining clauses then the Forward Chaining construction produces another set. That is $D^{\prec} = \{a, b, d\}$, $R^{\prec} = \{c\}$ and the clauses r_3

and r_5 become inconsistent. $\square$

It should be clear that this variant of the Forward Chaining construction generalizes the original Forward Chaining construction. In particular, all the sets $D^{\prec}$ constructed with the well-orderings of $nmon(P)$ can be constructed with the well-orderings of $P \setminus R$ (for any $R \subseteq mon(P)$). Specifically, the orderings of $P \setminus R$ in which $mon(P) \setminus R$ forms an initial segment have this property.

4 Complexity of Stable Submodels

In this section, we study the complexity of the Forward Chaining construction for finite and recursive programs. We begin this section with the basic definitions of recursive programs and recall some of the basic results on the complexity of stable models of recursive programs as proved in a series of our earlier papers [MNR90,MNR92c,MNR92a].

4.1 Preliminaries

Let ω denote the set of natural numbers. The canonical index, $can(X)$, of finite set $X = \{x_1 < \ldots < x_n\} \subseteq \omega$ is defined as $2^{x_1} + \ldots + 2^{x_n}$ and the canonical index of $\emptyset$ is defined as 0. Let D_k be the finite set whose canonical index is k, i.e., $can(D_k) = k$.

We shall identify a clause r with a triple $\langle k, l, \varphi \rangle$ where $D_k = prem(r)$, and $D_l = cons(r)$, $\varphi = c(r)$. In this way, when $H_P \subseteq \omega$ we can think about P as a subset of ω as well. This given, we then say that a program P is *recursive* if H_P and P are recursive subsets of ω.

Next we shall define various types of recursive trees and Π^0_1 classes. Let $[,] \colon \omega \times \omega \to \omega$ be a fixed one-to-one and onto recursive pairing function such that the projection functions π_1 and π_2 defined by $\pi_1([x, y]) = x$ and $\pi_2([x, y]) = y$ are also recursive. Extend our pairing function to code n-tuples for $n > 2$ by the usual inductive definition, that is, let $[x_1, \ldots, x_n] = [x_1, [x_2, \ldots, x_n]]$ for $n \geq 3$. Let $\omega^{<\omega}$ be the set of all finite sequences from ω and let $2^{<\omega}$ be the set of all finite sequences of 0's and 1's. Given $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $\beta = \langle \beta_1, \ldots, \beta_k \rangle$ in $\omega^{<\omega}$, write $\alpha \sqsubseteq \beta$ if α is initial segment of β, i.e. , if $n \leq k$ and $\alpha_i = \beta_i$ for $i \leq n$. In this paper, we identify each finite sequence $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ with its code $c(\alpha) = [n, [\alpha_1, \ldots, \alpha_n]]$ in ω. Let 0 be the code of the empty sequence $\emptyset$. When we say that a set $S \subseteq \omega^{<\omega}$ is recursive, recursively enumerable, etc., what we mean is that the set $\{c(\alpha) \colon \alpha \in S\}$ is recursive, recursively enumerable, etc. Define a *tree* T to be a nonempty subset of $\omega^{<\omega}$ such that T is closed under initial segments. Call a function $f \colon \omega \to \omega$ an infinite *path* through T provided that for all n, $\langle f(0), \ldots, f(n) \rangle \in T$. Let $[T]$ be the set of all infinite paths through T. Call a set A of functions a Π_1^0-class if there exists a recursive predicate R such that $A = \{f\colon \omega \to \omega : \forall n(R(n, [f(0), \dots, f(n)])\}$. Call a Π_1^0-class A *recursively bounded* if there exists a recursive function $g\colon \omega \to \omega$ such that $\forall f \in A \forall n(f(n) \leq g(n))$. It is not difficult to see that if A is a Π_1^0-class, then $A = [T]$ for some recursive tree $T \subseteq \omega^{<\omega}$. Say that a tree $T \subseteq \omega^{<\omega}$ is *highly recursive* if T is a recursive finitely branching tree and also there is a recursive procedure which, applied to $\alpha = \langle \alpha_1, \dots, \alpha_n \rangle$ in T, produces a canonical index of the set of immediate successors of α in T. Then if A is a recursively bounded Π_1^0-class, it is easy to show that $A = [T]$ for some highly recursive tree $T \subseteq \omega^{<\omega}$, see [JS72b]. For any set $A \subseteq \omega$, let $A' = \{e\colon \{e\}^A(e) \text{ is defined}\}$ be the jump of A, let $\mathbf{0}'$ denote the jump of the empty set $\emptyset$. We write $A \leq_T B$ if A is Turing reducible to B and $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$.

We say that there is an effective, one-to-one degree preserving correspondence between the set of stable models $\mathit{Stab}(P)$ of a recursive programs P and the set of infinite paths $[T]$ through a recursive tree T if there are indices e_1 and e_2 of oracle Turing machines such that
(i) $\forall_{f\in[T]}\{e_1\}^{gr(f)} = M_f \in \mathit{Stab}(P)$,
(ii) $\forall_{M\in \mathit{Stab}(P)}\{e_2\}^M = f_M \in [T]$, and
(iii) $\forall_{f\in[T]}\forall_{M\in \mathit{Stab}(P)}(\{e_1\}^{gr(f)} = M$ if and only if $\{e_2\}^M = f)$.
where $\{e\}^B$ denotes the function computed by the e^{th} oracle machine with oracle B. Also, write $\{e\}^B = A$ for a set A if $\{e\}^B$ is a characteristic function of A. For any function $f\colon \omega \to \omega$, $gr(f) = \{[x, f(x)]\colon x \in \omega\}$. Condition (i) says that the infinite paths of the tree T uniformly produce stable models via an algorithm with index e_1. Condition (ii) says that stable models of P uniformly produce infinite paths through T via an algorithm with index e_2. Condition (iii) asserts that if $\{e_1\}^{gr(f)} = M_f$, then f is Turing equivalent to M_f. In the sequel we shall not explicitly construct the indices e_1 and e_2, but it will be clear that such indices can be constructed in each case.

We shall use now the concept of a proof scheme (see Section 2) to define two important classes of programs. We depart from the fact that there is a natural preordering of proof schemes. We define, for proof schemes s_1, s_2, $s_1 \ll s_2$ if both s_1, s_2 have same conclusions and every clause appearing in s_1 appears in s_2. Although $\ll$ is not a partial ordering, it is well-founded. Thus for every proof scheme s there is a $\ll$-minimal proof scheme s_1 such that $s_1 \ll s$.

There are two important subclasses of recursive programs introduced in our paper [MNR92a], namely locally finite and highly recursive programs. Say that the program P is *locally finite* if for each $c \in H_P$, there exist only finitely many $\ll$-minimal proof schemes with conclusion c. If P is locally finite, then for every c, there exists a finite set of derivations Dr_c such that all the derivations of c are inessential variants of the derivations in Dr_c. That is, if p is a derivation of c, then there is a derivation $p_1 \in Dr_c$ such that $p_1 \ll p$. Finally, say that P is *highly recursive* if P is recursive, locally finite, and the map $c \mapsto can(Dr_c)$ is partial recursive. The latter means that there is an effective procedure which, when applied to any $c \in H_P$, produces a canonical index of the set of all $\ll$-minimal proof schema with conclusion c.

This given, we can now state some basic results from our earlier papers [MNR90,MNR92c,MNR92a] on the complexity of stable models of recursive programs.

Theorem 4.1 *For any highly recursive program P, there is a highly recursive tree T_P such that there is an effective 1:1 degree preserving correspondence between $[T_P]$ and Stab(P). Vice versa, for any highly recursive tree T, there is a highly recursive program P_T such that there is an effective 1:1 degree preserving correspondence between $[T]$ and Stab(P_T).*

Theorem 4.2 *For any locally finite recursive program P, there is a tree T_P which is highly recursive in $\mathbf{0}'$ such that there is an effective 1:1 degree preserving correspondence between $[T_P]$ and Stab(P). Vice versa, for any highly recursive tree T in $\mathbf{0}'$, there is a locally finite recursive program P_T such that there is an effective 1:1 degree preserving correspondence between $[T]$ and Stab(P_T).*

Theorem 4.3 *For any recursive program P, there is a recursive tree T_P such that there is an effective 1:1 degree preserving correspondence between $[T_P]$ and Stab(P). Vice versa, for any recursive tree T, there is a recursive program P_T such that there is an effective 1:1 degree preserving correspondence between $[T]$ and Stab(P_T).*

Because the set of degrees of paths through trees have been extensively studied in the literature, we immediately can derive a number of corollaries about the degrees of stable models in recursive programs. We shall give a few of these corollaries below.

For recursive programs, we have the following results, see [MNR92a].

Corollary 4.4 *(i) Every recursive program P, which has a stable model, has a stable model M such that $M \leq_T B$ where B is a complete Π^1_1-set.*

(ii) If P is a recursive program with a unique stable model M, then M is hyperarithmetic.

Corollary 4.5 *(i) There is a recursive program P such that P has a stable model but P has no stable model which is hyperarithmetic.*

(ii) For each recursive ordinal α, there exists a recursive program P possessing a unique stable model M such that $M \equiv_T \mathbf{0}^{(\alpha)}$.

These two corollaries show that the stable models of a recursive program may be very complex. We shall see in the next section that in contrast, there is always at least one stable submodel of a recursive program which occurs relatively low in the arithmetic hierarchy, namely there will always be a stable submodel which is r.e. in $\mathbf{0}'$.

We note that there are natural conditions which will guarantee that the set of stable models of a program are much better behaved. For example, if the program is highly recursive, then we have the following results, see [MNR92a].

Call A *low* if $A' \equiv_T \mathbf{0}'$. This means that A is low provided that the jump of A is as small as possible with respect to Turing degrees. The following corollary is an immediate consequence of Theorem 4.1 and the work of Jockusch and Soare [JS72b].

Corollary 4.6 *Let P be a highly recursive program such that $Stab(P) \neq \emptyset$. Then*
(i) There exists a stable model M of P such that M is low.
(ii) If P has only finitely many stable models, then every stable model M of P is recursive.

In the other directions, there are a number of corollaries of the Theorem 4.1 which allow us to show that there are highly recursive programs P such that the set of degrees realized by elements of *Stab*(P) are still quite complex. Again all these corollaries follow by transferring results of Jockusch and Soare [JS72b,JS72a].

Corollary 4.7 *(i) There is a highly recursive program P such that P has $2^{\aleph_0}$ stable models but no recursive stable models.*
(ii) There is a highly recursive program P such that P has $2^{\aleph_0}$ stable models and any two stable models $M_1 \neq M_2$ of P are Turing incomparable.
(iii) There is a highly recursive program P such that P has $2^{\aleph_0}$ stable models and if $\mathbf{a}$ is the degree of any stable model M of P and $\mathbf{b}$ is any recursively enumerable degree such that $\mathbf{a} <_T \mathbf{b}$, then $\mathbf{b} \equiv_T \mathbf{0}'$.
(iv) If $\mathbf{a}$ is any recursively enumerable Turing degree, then there is a highly recursive program P such that P has $2^{\aleph_0}$ stable models and the set of recursively enumerable degrees $\mathbf{b}$ which contain an stable model of P is precisely the set of all recursively enumerable degrees $\mathbf{b} \geq_T \mathbf{a}$.

Finally, we note that there are analogues of Corollaries 4.6 and 4.7 which hold for recursive locally finite general programs. That is, one can replace highly recursive general programs by recursive locally finite general programs if one replaces all the statements about degrees of stable models by the corresponding statement relative to an $\mathbf{0}'$ oracle. For example, the analogue of part (1) of Corollary 4.6 is that every recursive locally finite general program P such that *Stab*$(P) \neq \emptyset$ has a stable model M such that M is recursive in $\mathbf{0}''$, while the analogue of part (1) of Corollary 4.7 is that there exists a recursive locally finite general program P which has $2^{\aleph_0}$ stable models but which has no stable model which is recursive in $\mathbf{0}'$. See [MNR92c] for further details.

4.2 Complexity of the Forward Chaining Construction.

In this section we discuss complexity issues for sets of the form $D^{\prec}$, where P is a recursive program and $\prec$ is either some ordering of type ω or some finite ordering. First of all, recall that every stable model of P can be obtained as $D^{\prec}$ for a suitably chosen ordering $\prec$. This means that, since the stable models can be very complex, even if there is only one stable model, we cannot obtain results on complexity of $D^{\prec}$ without restricting the class of orderings. As noticed above there are recursive programs P such that P possess a unique stable model but that stable model is as high in the hyperarithmetical hierarchy as desired. Therefore we shall put now the restriction on the order type of $\prec$. This restriction is related to the fact that in any attempt to implement even a partial construction of $D^{\prec}$, we cannot go beyond ω. Moreover, ω (and finite ordinals) have the following property:

Lemma 4.8 *Let P be a program and let $\prec$ be a well-ordering of nmon(P) of order type $\leq \omega$. Then the closure ordinal of the construction of the family $\langle D^{\prec}_{\xi} \rangle$ is at most ω.*

Proof: Our lemma is obvious for the case of finite *nmon*(P). Hence assume that *nmon*(P) is infinite. If the closure ordinal of the construction of $D^{\prec}$ is greater than ω, then $D^{\prec}_{\omega+1} \neq D^{\prec}_{\omega}$ and in particular r_ω is defined. Then $c(r_\omega) \notin D^{\prec}_{\omega}$, $prem(r_\omega) \subseteq D^{\prec}_{\omega}$, $(cons(r_\omega) \cup R^{\prec}_{\omega}) \cap (cl_{\mathrm{mon}}(D^{\prec}_{\omega} \cup \{c(r_\omega)\}) = \emptyset$. Let $\{s_k : k \in \omega\}$ be the enumeration of *nmon*(P) in the order $\prec$. Then for some $k \in \omega$, $r_\omega = s_k$. Since k is finite, there must be a natural number l satisfying the following conditions:

(i) $prem(r_\omega) \subseteq D^{\prec}_{l}$;
(ii) For all $j < k$ if $c(s_j) \in D^{\prec}_{\omega}$, then $c(s_j) \in D^{\prec}_{l}$.

Selecting least l with these properties we see that $r_{l+1} = r_\omega$, which is a contradiction. □

It is easy to see that the property indicated in Lemma 4.8 does not hold for ordinals greater than ω.

Example 4.1 Let $H_P = \{a_n : n \in \omega\} \cup \{b_n : n \in \omega\} \cup \{c, d\}$, let $P = \{C_0, C_1, E_0, E_1, E_2, \ldots\}$ where $C_0 = c \leftarrow \neg d$, $C_1 = a_1 \leftarrow c, \neg d$, $E_0 = a_0 \leftarrow$ and $E_m = a_{m+2} \leftarrow a_m, \neg b_m$, $m \geq 1$. Let $\prec$ be the ordering on *nmon*(P) defined by $r_1 \prec r_2 \prec \ldots \prec q_0 \prec q_1$. Then $\prec$ is an ordering of the type $\omega + 2$. But it is easy to see that with this ordering the construction of $D^{\prec}$ will take precisely $\omega + \omega$ steps. In the first step we compute a_0, then at the step n we add a_{2n}. At the step ω we add c, at the step $\omega + 1$, a_1 and at each step $\omega + n$, a_{2n+1}. Finally, at the stage $\omega + \omega$ construction closes. □

We shall restrict our attention now to the case when P is recursive and $\prec$ is a recursive well-ordering of type ω.

Proposition 4.9 *Let P be a recursive general program. Let $\prec$ be a recursive well-ordering of nmon(P) of order type $\leq \omega$. Finally, let $D^{\prec}, R^{\prec}, I^{\prec}$, and $A^{\prec}$ be sets of atoms and of clauses defined in Definition 3.1. Then:*

(i) $D^{\prec}$ *is r.e. in* $\mathbf{0}'$.
(ii) $R^{\prec}$ *is r.e. in* $\mathbf{0}'$.
(iii) $I^{\prec}$ *is recursive in* $\mathbf{0}''$.
(iv) $A^{\prec}$ *is recursive in* $\mathbf{0}''$.

Proof: Clearly, $mon(P)$ and $nmon(P)$ are recursive sets. Since $mon(P)$ is recursive it follows that for any r.e. set $M \subseteq H_P$, $cl_{\text{mon}}(M)$ is also r.e. In fact it is uniform. i.e. we can find a recursive function f such that for a set W_e, $cl_{\text{mon}}(W_e) = W_{f(e)}$.

The closure ordinal of the construction of the sets $D_{\xi}^{\prec}$ (and thus $R_{\xi}^{\prec}$) is at most ω (by Lemma 4.8). It is easy to see that each $R_i^{\prec}$ is finite. As concerns $D_i^{\prec}$, each of these sets is r.e., as is easily proved by induction.

If we end up in a finite number of steps, say n, then it follows that $D^{\prec}$ is r.e. and $R^{\prec}$ is finite. Thus certainly $D^{\prec}$ is r.e. in $\mathbf{0}'$, and similarly for $R^{\prec}$.

If the closure ordinal of our construction is exactly ω, we are dealing with two sequences:

$$D_0^{\prec} \subseteq D_1^{\prec} \subseteq \ldots$$

$$R_0^{\prec} \subseteq R_1^{\prec} \subseteq \ldots$$

The first sequence consists of r.e. sets, the second of finite sets. Now we want to evaluate the complexity of union of each hierarchy. To this end let us notice that with an $\mathbf{0}'$ oracle, we can effectively find r_n, and so we have a function h, recursive in $\mathbf{0}'$, such that $D_n^{\prec} = W_{h(n)}$. Then, we can write:

$$x \in D^{\prec} \equiv \exists_n x \in W_{h(n)}$$

or equivalently

$$x \in D^{\prec} \equiv \exists_n \exists_k (k = h(n) \wedge x \in W_k)$$

Since h is recursive in $\mathbf{0}'$, $D^{\prec}$ is r.e. in $\mathbf{0}'$.

A similar argument is used for item (2), except that instead of W_e we consider an enumeration of finite sets. The complexity does not lower, since the existential quantifier in front is still there, and the function that produces the canonical index of $R_{n+1}^{\prec}$ out of $R_n^{\prec}$ is recursive in $\mathbf{0}'$.

Finally, given the $\mathbf{0}''$ oracle we can decide the question of membership of elements in sets r.e. in $\mathbf{0}'$. In particular we can decide if $prem(r) \subseteq D^{\prec}$, and if $cl_{\text{mon}}(\{c(r)\} \cup D^{\prec}) \cap (cons(r) \cup R^{\prec}) \neq \emptyset$. Thus $I^{\prec}$ is recursive in $\mathbf{0}''$, and hence $A^{\prec}$ is recursive in $\mathbf{0}''$ as well. □

Corollary 4.10 *If P is a recursive program such that $nmon(P)$ is finite, then for any ordering $\prec$ of $nmon(P)$, $D^{\prec}$ is r.e., $R^{\prec}$ is finite, and $I^{\prec}$ is finite and $A^{\prec}$ is recursive.* □

We shall now define a class of programs for which stronger results can be obtained.

Definition 4.11 Let P be a recursive program. We say that P is *monotonically decidable* if

(i) For every finite $A \subseteq H_P$, $cl_{\text{mon}}(A)$ is recursive;
(ii) There is a recursive function f such that given a canonical index k, $f(k)$ is an index of a partial recursive function φ_k such that φ_k is the characteristic function of $cl_{\text{mon}}(D_k)$.

Notice that this concept depends only on the monotonic part of the program P. The idea here is that we can find the characteristic function of $cl_{\text{mon}}(A)$ uniformly in A. For example, if $mon(P)$ is finite, then P is monotonically decidable.

If the program P is monotonically decidable we can strengthen Proposition 4.9 considerably.

Proposition 4.12 *If P is a recursive monotonically decidable program and $\prec$ is a recursive ordering of $nmon(P)$ of type $\leq \omega$, then*

(i) If $nmon(P)$ is finite, then $D^{\prec}$ is recursive and $A^{\prec}$ is recursive (and $I^{\prec}$ and $R^{\prec}$ are finite).
(ii) If $nmon(P)$ is infinite, then $D^{\prec}$ and $R^{\prec}$ are r.e. and $I^{\prec}$ and $A^{\prec}$ are recursive in $\mathbf{0}'$.

Proof: Since P is monotonically decidable, $D_n^{\prec}$ is recursive for every n. Indeed, the operator cl_{mon} is monotone, finitizable and idempotent and therefore $D_n^{\prec} = cl_{\text{mon}}(\{c(r_j) : j \leq n, r_j \text{ defined}\}$.

Moreover, our assumption that $\prec$ is effective means that we can search the list $nmon(P)$ effectively to see if there is a clause r such that $prem(r) \subseteq D_n^{\prec}$, $(cons(r) \cup \{c(r)\}) \cap D_n^{\prec} = \emptyset$ and $cl_{\text{mon}}(D_n^{\prec} \cup \{c(r)\}) \cap (cons(r) \cup R_n^{\prec}) = \emptyset$. This implies that if r_{n+1} exists we can effectively find it.

Thus, as in the proof of Proposition 4.9 either for some n, $D_n^{\prec} = D_{n+1}^{\prec}$ and so $D_n^{\prec} = D^{\prec}$ (and in this case $D^{\prec}$ is recursive and $R^{\prec}$ is finite) or the construction closes at ω and we are dealing with an effective sequence of recursive sets

$$D_0^{\prec} \subseteq D_1^{\prec} \subseteq D_2^{\prec} \ldots$$

and an effective sequence of finite sets

$$R_0^{\prec} \subseteq R_1^{\prec} \subseteq R_2^{\prec} \ldots$$

Proceeding similarly as in the proof of Proposition 4.9 we establish that $D^{\prec}$ is r.e. and that $R^{\prec}$ is r.e.

Again, reasoning as in Proposition 4.9 (except that an oracle for $\mathbf{0}'$ is now enough) one can easily show that $I^{\prec}$ and $A^{\prec}$ are recursive in $\mathbf{0}'$.

If $nmon(P)$ is finite then the construction must stop in a finite number of steps and the first case applies. Since $I^{\prec} \subseteq nmon(P)$, $I^{\prec}$ is a finite set, and so $A^{\prec}$ is recursive. $\Box$

Now let us look at the case of finite P. In our complexity considerations, every atom a will have the cost $||a||$. Next, for a clause

$$r = c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$$

we define $||r|| = (\sum_{i \leq n} ||a_i||) + (\sum_{i \leq m} ||b_j||) + ||c||$. Finally, for a set Q of clauses we define

$$||Q|| = \sum_{r \in Q} ||r||.$$

Theorem 4.13 *Suppose P is a finite general program and $\prec$ is some well-ordering of $nmon(P)$. Then $D^{\prec}, R^{\prec}, A^{\prec}$, and $I^{\prec}$ can be computed in time $O(||mon(P)||\,||nmon(P)|| + ||nmon(P)||^2)$.*

Proof: First consider a stage $k+1$ in the Forward Chaining construction. Given $D_k^{\prec}$ and $R_k^{\prec}$, we must make a pass in order through the clauses to check for each clause $r = c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$ whether

$$\{a_1, \ldots, a_n\} \subseteq D_k^{\prec} \text{ and } \{b_1, \ldots, b_m, c\} \cap D_k^{\prec} = \emptyset.$$

Notice that at a cost of maintaining an appropriate data structure we can perform this check in $C||r||$ steps for some constant C and we call such a check a *clause check*. Now if $\{a_1, \ldots, a_n\} \subseteq D_k^{\prec}$ and $\{b_1, \ldots, b_m, c\} \cap D_k^{\prec} = \emptyset$, then we must compute $cl_{\text{mon}}(\{c\} \cup D_k^{\prec})$ and check whether $cl_{\text{mon}}(\{c\} \cup D_k^{\prec}) \cap (\{b_1, \ldots, b_m\} \cup R_k^{\prec}) = \emptyset$. Now assuming that we process the clauses in order, if $cl_{\text{mon}}(\{c\} \cup D_k^{\prec}) \cap (\{b_1, \ldots, b_m\} \cup R_k^{\prec}) = \emptyset$, then $r = r_{k+1}$, $D_{k+1}^{\prec} = cl_{\text{mon}}(\{c\} \cup D_k^{\prec})$, and $R_{k+1}^{\prec} = \{b_1, \ldots, b_m\} \cup R_k^{\prec}$. If $cl_{\text{mon}}(\{c\} \cup D_k^{\prec}) \cap (\{b_1, \ldots, b_m\} \cup R_k^{\prec}) \neq \emptyset$, then we know that r can never be a candidate to r_j for any $j > k$ so that we can just mark clause r and never consider it again. Of course we also mark r_{k+1} at stage $k+1$ if it is defined so that at each stage we will mark at least one $r \in nmon(P)$. Moreover, if r_{k+1} is not defined, then we can stop since then we know $D_k^{\prec} = D^{\prec}$ and $R_k^{\prec} = R^{\prec}$.

It follows that at stage $k+1$, we need to look at most $|nmon(P)|$ - k clauses and hence perform at most $|nmon(P)|$ -k clause checks. Since the construction must stop at stage *nmon*(P). it follows that the entire construction requires at most

(a) $\begin{pmatrix} |nmon(P)| \\ 2 \end{pmatrix}$ clause checks,

(b) $|nmon(P)|$ operations of computing $cl_{\mathrm{mon}}(D_k^{\prec} \cup \{c\})$ and checking if $cl_{\mathrm{mon}}(D_k^{\prec} \cup \{c\}) \cap (\{b_1, \ldots b_m\} \cup R_k^{\prec}) = \emptyset$ and

(c) the computation of $cl_{\mathrm{mon}}(\emptyset)$.

Since $\binom{x}{y} = 1/2x(x+1)$, consequently clause checks require $O(||nmon(P)||^2)$ steps. Next consider the computations of $cl_{\mathrm{mon}}(A)$ and the checking of whether $cl_{\mathrm{mon}}(A) \cap B = \emptyset$ for $A, B \subseteq H_P$ where $A \cap B = \emptyset$ which are required for **(b)** and **(c)** above. We claim all this can be done in time proportional to $O(||mon(P)||\,||nmon(P)||)$. Since in our construction all the elements of $D_k^{\prec}$ and $R_k^{\prec}$ must appear in one of the clauses, we can assume $||A|| + ||B|| \leq ||mon(P)|| + ||nmon(P)||$. Now we can first make a pass through all the clauses of *nmon*(P) to get a list of all the elements of H_P which occur in one of the clauses. Call this set V. Another pass through the clauses will allow us to set up a system of pointers from each $c \in V$ to the set of clauses $r \in mon(P)$ such that c occurs in the set of premises of r. We can also mark which c are in A and which c are in B. All this will require $O(||mon(P)|| + ||nmon(P)||) \leq O(||mon(P)||\,||nmon(P)||)$ steps. Now for each $c \in A$, use the pointers from c to the clauses $r \in mon(P)$ to update each r by marking each premise of r in A. Now if a clause $r \in mon(P)$ has all of its premises marked, we mark the conclusion of r, *i.e.*, we add the $cnl(r)$ to $cl_{\mathrm{mon}}(A)$, and use the pointers from $cln(r)$ to clauses in *mon*(P) to further update the premises of each clause by marking $cln(r)$. We continue in this fashion until there are no more clauses to update in which case A together with the marked conclusions will form the $cl_{\mathrm{mon}}(A)$. Of course, if an element of B turns up, we stop the construction. Thus either we will find that $cl_{\mathrm{mon}}(A) \cap B \neq \emptyset$ or we will complete the computation of $cl_{\mathrm{mon}}(A)$ and be assured that $cl_{\mathrm{mon}}(A) \cap B \neq \emptyset$. Now assuming that updates can be performed in constant time, each clause $r \in mon(P)$ can require at most $||r||$ updates in this process since once all the premises of a clause have been marked we no longer have to consider it. Thus we require at most $||mon(P)||$ updates so the entire process takes at most $O(||mon(P)||)$ steps. Thus to compute the monotonic closures and intersection checks required in **(b)** and **(c)** above takes $O((1 + ||mon(P)||)||nmon(P)||) \leq O(||mon(P)||\,||nmon(P)||)$ steps. □

5 Variants of the Forward Chaining Construction

In this section, we briefly explore the possibility of simplifying our Forward Chaining construction. First we show that it is possible to reconstruct all stable models of a program via a Forward Chaining algorithm which requires that we neither monotonically close at each stage nor do we have to check consistency. As we shall see a drawback of this type of construction is that when it does not work, we may not even get a stable submodel. Then we shall give the definition of FC-normal programs as defined in [MNR93b]. FC-Normal programs have the property that all stable submodels are in fact stable models and moreover we can drop the consistency check in the Forward Chaining algorithm.

5.1 *Computing Extensions without Maintaining Consistency*

We begin by giving a simplified variant of the construction given in Section 3 which does monotonically close at each stage and which does not check consistency.

Let P be a general program. Let $\prec$ be a well-ordering of P. We define a sequence of subsets of subsets of H_P, $\langle M_\xi \rangle_{\xi<\alpha}$, a sequence of elements of P, $\langle d_\xi \rangle_{0<\xi<\alpha}$, a sequence of elements of H_P, $\langle a_\xi \rangle_{0<\xi<\alpha}$, and an ordinal $\alpha = \alpha_\prec$ inductively as follows.

$M_0 = \emptyset$ (notice that d_0 and a_0 are not defined at all). Assume that $\langle M_\xi \rangle_{\xi<\beta}$, $\langle d_\xi \rangle_{0<\xi<\beta}$, $\langle a_\xi \rangle_{0<\xi<\beta}$ have been defined but α has not been defined. If there is no clause $r \in P \setminus \{d_\xi : 0 < \xi < \beta\}$ such that $prem(r) \subseteq \bigcup_{\xi<\beta} M_\xi$, $cons(r) \cap \bigcup_{\xi<\beta} M_\xi = \emptyset$, then $\alpha = \beta$ and the construction is completed.

Otherwise, d_β is the $\prec$-first clause in $r \in P \setminus \{d_\xi : 0 < \xi < \beta\}$ such that $prem(r) \subseteq \bigcup_{\xi<\beta} M_\xi$, $cons(r) \cap \bigcup_{\xi<\beta} M_\xi = \emptyset$ and $a_\beta = c(d_\beta)$ (and α is not yet defined). Finally, $M_\beta = \bigcup_{\xi<\beta} M_\xi \cup \{a_\beta\}$.

Clearly, because of cardinality argument, there is β such that d_β is not defined. Such $\beta < |P|^+$. Therefore α is defined.

Given a general program P and a well-ordering $\prec$ of P define $M_\prec = \bigcup_{\xi<\alpha} M_\xi$, $D_\prec = \{d_\xi : 0 < \xi < \alpha\}$. The following two propositions have been proved for the case of default logic in [MT93]

Proposition 5.1 *If $cons(D_\prec) \cap M_\prec = \emptyset$, then $M_\prec$ is a stable model of P.*

Proof: Let $A = M_\prec$, $D = D_\prec$. Assume $cons(D) \cap A = \emptyset$. By an easy induction on ξ we prove that a_ξ, belongs to $C_A(\emptyset)$ for all $\xi < \alpha$. Hence $A \subseteq C_A(\emptyset)$.

For the other inclusion, assume $C_A(\emptyset) \setminus A \neq \emptyset$. Then consider the element in $C_A(\emptyset) \setminus A$ with the shortest possible derivation. There must exist a clause $r \in P$ in such a derivation such that $prem(r) \subseteq A$, $cons(r) \cap A = \emptyset$, and $c(r) \notin A$. In particular $c(r) \neq a_\xi$ for all $\xi < \alpha$. Thus $r \neq d_\xi$ for all $\xi < \alpha$. But then the $\prec$-first such clause can serve as a definition of a_α, which contradicts our definition of α. □

Conversely, every stable model of P is of the form $M_\prec$ for a suitably chosen

$\prec$.

Proposition 5.2 *If M is a stable model of P, then for some $\prec$, $M = M_\prec$.*

Proof: Choose $\prec$ in such a way that $G(M, P)$ forms an initial segment of $\prec$. We claim that $M_\prec = M$.

By induction on $\xi < \alpha$ we show that $a_\xi \in M$. Indeed. assume that for all $\xi < \beta$, $a_\xi \in M$. If d_ξ is not defined then $\beta = \alpha$ and so, by inductive assumption, all a_ξ belong to M. If d_ξ is defined, then there is a clause r such that $r \neq d_\xi$ for all $\xi < \beta$, $prem(r) \subseteq \bigcup_{\xi<\beta} M_\xi$, $cons(r) \cap \bigcup_{\xi<\beta} M_\xi = \emptyset$. Select the $\prec$-first such clause r. We prove that $r \in G(M, P)$.

Since $\{a_\xi : \xi < \beta\} \subseteq M$ two cases are possible.
(a) $\{a_\xi : \xi < \beta\} = M$. Then $r \in G(M, P)$ by definition.
(b) $\{a_\xi : \xi < \beta\} \subset M$. Then, for some $j < \omega$ all elements with M-derivation of length smaller than j are in $\{a_\xi : 0 < \xi < \beta\}$ but some element with an M-derivation of the length j is not in $\{a_\xi : 0 < \xi < \beta\}$. But then there must be a clause r' in such a derivation such that $prem(r') \subseteq \{a_\xi : 0 < \xi < \beta\}$ and $cons(r') \cap M = \emptyset$. In particular $r' \in G(M, P)$. Since $r \prec r'$ or $r = r'$, $r' \in G(M, P)$, and $G(M, P)$ forms an initial segment of $\prec$, $r \in G(M, P)$. Since $d_\beta = r$, $a_\beta = c(r)$. But M is a stable model, thus a supported model of P. Hence $a_\beta \in M$. Hence we proved that $M_\prec \subseteq M$.

To show the converse inclusion we show, as above, that if $M_\prec \setminus M \neq \emptyset$. then d_α can be defined, contradicting the choice of α. Thus $M = M_\prec$. □

Proposition 5.2 shows that any given stable model of the program can be constructed via our simplified Forward Chaining construction for some ordering. However arbitrary outputs of this simplified Forward Chaining construction may not have any nice properties. That is, the output of our simplified Forward Chaining construction may not be a stable model of any subprogram of the original program. Essentially what can happen is that our simplified version of the Forward Chaining construction allows us to produce sets which are too large to be a stable model of any subprogram of our original program. This is illustrated in the example below.

Example 5.1 Let $H_P = \{a, b, c, d, e\}$ and let P consist of the following clauses:

(i) $a \leftarrow$
(ii) $e \leftarrow b$
(iii) $c \leftarrow d$
(iv) $d \leftarrow a, \neg b$
(v) $b \leftarrow a, \neg d$
(vi) $b \leftarrow c, \neg e$

Here, $mon(P)$ consists of clauses (1), (2) and (3), whereas $nmon(P)$ consists of clauses (4), (5), and (6).

We note that this program is just a subprogram of the program of Example 3.2 and does have a stable model. Indeed in Example 3.2 we showed that the Forward Chaining construction produces $D^{\prec'} = \{a, b, e\}$ when we assume (6)$\prec'$(5)$\prec'$(4) which is a stable model of P. One can easily check that the simplified version of the Forward Chaining construction applied to the order $\prec'$, (6)$\prec'$(5)$\prec'$(4)$\prec'$(3)$\prec'$(2)$\prec'$(1) will produce the same set, i.e. $M_{\prec'} = \{a, b, e\}$.

Next consider the order

$$(1) \prec (2) \prec (3) \prec (4) \prec (5) \prec (6).$$

Then the simplified version of Forward Chaining construction produces something very different. That is, the stages of the simplified version of the Forward Chaining construction are as follows:

Stage 0 $M_0 = \emptyset$.
Stage 1 $d_1 = (1)$, $a_1 = a$, and $M_1 = \{a\}$.
Stage 2 $d_2 = (4)$, $a_2 = d$, and $M_2 = \{a, d\}$.
Stage 3 $d_3 = (3)$, $a_3 = c$, and $M_3 = \{a, c, d\}$.
Stage 4 $d_4 = (6)$, $a_4 = b$, and $M_4 = \{a, b, c, d\}$.
Stage 5 $d_5 = (2)$, $a_5 = e$, and $M_5 = \{a, b, c, d, e\}$.

Thus $M_{\prec} = \{a, b, c, d, e\}$ which is not a stable model of any subprogram of P.

However for the order $(4) \prec (5) \prec (6)$, the Forward Chaining construction of $D^{\prec}$ produces these stages:
Stage 0 $D_0^{\prec} = cl_{\mathrm{mon}}(\emptyset) = \{a\}$, $R_0^{\prec} = \emptyset$.
Stage 1 $r_1 = (4)$, $D_1^{\prec} = cl_{\mathrm{mon}}(\{a\} \cup \{d\}) = \{a, c, d\}$, $R_1^{\prec} = \{b\}$.
Stage 2 r_2 is undefined and the construction stops at this stage. Hence $D^{\prec} = D_1^{\prec} = \{a, c, d\}$. We check that $I^{\prec'}$ consists of the clause (6). $j^{\prec'}$ consists of clauses (1)-(5). Thus $\{a, c, d\}$ is a stable model of the program $(1) - (5)$ but not of $(1) - (6)$. □

In fact the same relative order on the nonmonotonic clauses of the program can lead to completely different results under the two constructions. This is illustrated in our next example where the Forward Chaining construction produces a stable model while the simplified Forward Chaining construction produces a set which is not a stable model of any subprogram.

Example 5.2 Let $H_P = \{a, b, c, d, e\}$ and let P consist of the following clauses:

(i) $a \leftarrow$
(ii) $e \leftarrow b$
(iii) $c \leftarrow d$
(iv) $c \leftarrow e$
(v) $d \leftarrow a, \neg b$
(vi) $b \leftarrow a, \neg d$

(vii) $d \leftarrow a, \neg c$
(viii) $b \leftarrow c, \neg e$

Here, $mon(P)$ consists of clauses (1), (2), (3) and (4), whereas $nmon(P)$ consists of clauses (5). (6), (7) and (8).

Consider the order

$$(8) \prec (7) \prec (6) \prec (5) \prec (4) \prec (3) \prec (2) \prec (1).$$

The stages of the simplified version of the Forward Chaining construction are as follows:

Stage 0 $M_0 = \emptyset$.
Stage 1 $d_1 = (1)$, $a_1 = a$, and $M_1 = \{a\}$.
Stage 2 $d_2 = (7)$, $a_2 = d$, and $M_2 = \{a, d\}$.
Stage 3 $d_3 = (5)$, $a_3 = d$, and $M_3 = \{a, d\}$.
Stage 4 $d_4 = (3)$, $a_4 = c$, and $M_4 = \{a, c, d\}$.
Stage 5 $d_5 = (8)$, $a_5 = b$, and $M_5 = \{a, b, c, d\}$.
Stage 6 $d_6 = (2)$, $a_6 = e$, and $M_6 = \{a, b, c, d, e\}$.
Stage 7 $d_7 = (4)$, $a_6 = c$, and $M_7 = \{a, b, c, d, e\}$.

$M_\prec = \{a, b, c, d, e\}$ which is not a stable model of any subprogram of P.

However for the order $(8) \prec (7) \prec (6) \prec (5)$, the Forward Chaining construction of $D^\prec$ produces these stages:
Stage 0 $D_0^\prec = cl_{\text{mon}}(\emptyset) = \{a\}$, $R_0^\prec = \emptyset$.
Stage 1 $r_1 = (6)$, $D_1^\prec = cl_{\text{mon}}(\{a\} \cup \{b\}) = \{a, b, c, e\}$, $R_1^\prec = \{d\}$.
Stage 2 r_2 is undefined and the construction stops at this stage. Hence $D^\prec = D_1^\prec = \{a, b, c, e\}$. We check that $I^\prec = \emptyset$ so that $D^\prec$ is a stable model of P. Note that clause (7) is not applicable at Stage 1 of the Forward Chaining construction because $cl_{\text{mon}}(\{a\} \cup \{d\}) = \{a, c, d\}$ which contains the constraint of clause (7). □

5.2 FC-Normal Programs

In this section we shall define FC-normal programs and state the basic results about such programs proved in [MNR93b]. We shall see that FC-normal programs have the property that the Forward Chaining construction always produces a stable model. In fact for FC-normal programs, one can drop the consistency check in the Forward Chaining construction and it will still always produce a stable model.

Definition 5.3 Let P be a program. We say that a subset $Con \subseteq \mathcal{P}(H_P)$ (where $\mathcal{P}(H_P)$ is the power set of H_P) is a *consistency property* over P if

(i) $\emptyset \in Con$,
(ii) $\forall_{A,B \subseteq H_P}(A \subseteq B \quad \& \quad Con(B) \Rightarrow Con(A))$.
(iii) $\forall A_{\subseteq H_P}(Con(A) \Rightarrow Con(cl_{mon}(A)))$, and
(iv) whenever $\mathcal{A} \subseteq Con$ has the property that $A, B \in \mathcal{A} \rightarrow \exists_{C \in \mathcal{A}}(A \subseteq C \wedge B \subseteq C)$, then $Con(\bigcup \mathcal{A})$.

Condition (1) says that the empty set is consistent. Condition (2) requires that a subset of a consistent set is also consistent. Condition (3) postulates that the closure of a consistent set under Horn clauses of the program is consistent. Finally, the last condition says that the union of a *directed* family of consistent sets is also consistent. We note that conditions (1),(2), and (4) are Scott's conditions for information systems. Condition (3) connects "consistent" sets to the Horn part of the program; if A is consistent then adding elements derivable from A via Horn clauses preserves "consistency".

Definition 5.4 Let P be a program and let Con be a consistency property over P.

(i) A clause $C = c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_k \in nmon(P)$ is *FC-normal* (with respect to Con) if $Con(V \cup \{c\})$ and not $Con(V \cup \{c, b_i\})$ for all $i \leq k$ whenever $V \subseteq H_P$ is such that $Con(V)$, $cl_{mon}(V) = V$, $a_1, \ldots, a_n \in V$, and $c, b_1, \ldots, b_k \notin V$.
(ii) P is a *FC-normal* (with respect to Con) program if all $r \in nmon(P)$ are FC-normal with respect to Con.
(iii) P is *FC-normal program* if for some property $Con \subseteq \mathcal{P}(H_P)$. P is FC-normal with respect to Con.

Example 5.3 Let $H_P = \{a, b, c, d, e, f\}$. Let the consistency property be defined by the following condition:
$A \notin Con$ if and only if either $\{c, d\} \subseteq A$ or $\{e, f\} \subseteq A$.
Thus $\{a, b, c, e\}$, $\{a, b, c, f\}$, $\{a, b, d, e\}$, and $\{a, b, d, f\}$ are the maximal subsets of $\mathcal{P}(H_P)$ which are in Con.

Now consider the following program, P:

(1) $a \leftarrow$
(2) $b \leftarrow c$
(3) $c \leftarrow b$
(4) $c \leftarrow a, \neg d$
(5) $e \leftarrow c, \neg f$

Then for the program P, clauses (1),(2), and (3) form the monotonic (Horn) part of P and clauses (4) and (5) form the nonmonotonic part of P. First it is easy to check that Con is a consistency property over P. The monotonically closed subsets of $\mathcal{P}(H_P)$ which are in Con are the following sets: $\{a\}$,$\{a, d\}$,$\{a, e\}$,$\{a, f\}$,$\{a, b, c\}$, $\{a, d, e\}$, $\{a, d, f\}$, $\{a, b, c, e\}$,and $\{a, b, c, f\}$. It is then easy to check that P is FC-normal with respect to Con. Moreover one can easily check that P has a unique stable model $M = \{a, b, c, e\}$.

If we add to P the clause $d \leftarrow c$ to get a program P', then Con is no longer a consistency property over P' because $\{c\} \in Con$ but the monotonic closure of $\{c\}$ relative to P' which equals $\{a, b, c, d\}$ is not in Con.

If we add the clause $d \leftarrow e, \neg f$ to P to form a new program P'', Con will still be a consistency property over P'' because the property of being a consistency property depends only on the Horn part of the program. However P'' is not FC-normal with respect to Con because $r = d \leftarrow e, \neg f$ is not FC-normal with respect to Con. That is, for the monotonically closed set $\{a, b, c, e\}$, we have $prem(r) \subseteq \{a, b, c, e\}$, $cons(r) \cap \{a, b, c, e\} = \emptyset$, but $cl_{mon}(\{c(r)\} \cup \{a, b, c, e\}) = \{a, b, c, d, e\}$ is not in Con.

Finally if we add to P the clause $f \leftarrow c, \neg e$ then the resulting program P''' is still FC-normal with respect to Con but now there are two stable models, $M_1 = \{a, b, c, e\}$ and $M_2 = \{a, b, c, f\}$. □

FC-normal programs have all the desirable properties that are possessed by normal default theories as defined by Reiter in [Rei80]. In fact, it is shown in [MNR93b] that when one translates FC-normal programs back into the language of default logics than one obtains a class of default theories called *extended FC-normal* default theories which properly contains all normal default theories. We next shall state the basic results about FC-normal programs from [MNR93b].

Theorem 5.5 *Let P be a FC-normal program then there exists a stable model of P.*

Theorem 5.6 *Let P be a FC-normal program with respect to consistency property Con and let I be a subset of H_P such that $I \in Con$. Then there exists a stable model M of P such that $I \subseteq M$.*

In fact all stable models of FC-normal programs can be constructed via a slightly simplified version of the Forward Chaining construction which we shall call the Normal Forward Chaining construction. To this end, fix some well-ordering $\prec$ of $nmon(P)$. That is, the well-ordering $\prec$ determines some listing of the clauses of $nmon(P)$,$\{r_\alpha : \alpha \in \gamma\}$ where γ is some ordinal. Let Θ_γ be the least cardinal such that $\gamma \leq \Theta_\gamma$. In what follows, we shall assume that the ordering among ordinals is given by $\in$. Our normal Forward Chaining construction will define an increasing sequence of sets $\{M_\alpha^\prec\}_{\alpha \in \Theta_\gamma}$. We will then define $M^\prec = \cup_{\alpha \in \Theta_\gamma} M_\alpha^\prec$. In [MNR93b] it is shown that $M^\prec$ is always an stable model of P. Moreover it is shown in [MNR93b] that all stable models of P arise from this construction.

The Normal Forward Chaining construction of $M^\prec$.

Case 0. Let $M_0^\prec = cl_{mon}(\emptyset)$.

Case 1. $\alpha = \eta + 1$ is a successor ordinal.
Given $M_\eta^\prec$, let $\ell(\alpha)$ be the least $\lambda \in \gamma$ such that

$$r_\lambda = s \leftarrow a_1, \ldots, a_p, \neg b_1, \ldots . \neg b_k$$

where $a_1, \ldots, a_p \in M_\eta^\prec$ and $b_1, \ldots, b_k$, $s \notin M_\eta^\prec$. If there is no such $\ell(\alpha)$, then let $M_{\eta+1}^\prec = M_\alpha^\prec = M_\eta^\prec$. Otherwise, let

$$M_{\eta+1}^\prec = M_\alpha^\prec = cl_{mon}(M_\eta^\prec \cup \{cln(r_{\ell(\alpha)})\}).$$

Case 2. α is a limit ordinal. Then let $M_\alpha^\prec = \bigcup_{\beta \in \alpha} M_\beta^\prec$.

This given, we have the following.

Corollary 5.7 *If P is a FC-normal program and $\prec$ is* any *well-ordering of $nmon(p)$, then*

(i) $M^\prec$ is a stable model of P.
(ii) (Completeness of the construction). Every stable model of P is of the form $M^\prec$ for a suitably chosen ordering $\prec$ of $nmon(P)$.

It is quite straightforward to prove by induction that if P is FC-normal with respect to consistency property Con, then $M_\alpha^\prec \in Con$ for all α and hence $M^\prec \in Con$. Thus the following is an immediate consequence of Theorem 5.7(2).

Corollary 5.8 *Let P be a FC-normal program with respect to consistency property Con, then every stable model of P is in Con.*

Example 5.4 If we consider the final extended program of Example 5.3, it is easy to check that any ordering $\prec_1$ in which the clause $r_1 = e \leftarrow c, \neg f$ precedes the clause $r_2 = f \leftarrow c, \neg e$ will have $M^{\prec_1} = M_1$ while any ordering $\prec_2$ in which r_2 precedes r_1 will have $M^{\prec_2} = M_2$. □

We should also point out that if we restrict ourselves to countable programs P, i.e. if H_P is countable, then we can restrict ourselves to orderings of order type ω where ω is the order type of the natural numbers. That is, suppose we fix some well-ordering $\prec$ of $nmon(P)$ of order type ω. Thus, the well-ordering $\prec$ determines some listing of the clauses of $nmon(P)$,$\{r_n : n \in \omega\}$. Our normal Forward Chaining construction can be presented in an even more straightforward manner in this case. Our construction again will define an increasing sequence of sets $\{M_n^\prec\}_{n\in\omega}$ in stages. This given, we will then define $M^\prec = \cup_{n\in\omega} M_n^\prec$.

The Countable Normal Forward Chaining construction of $M^\prec$.

Stage 0. Let $M_0^\prec = cl_{mon}(\emptyset)$.

Stage $n+1$. Let $\ell(n+1)$ be the least $s \in \omega$ such that

$$r_s = t \leftarrow a_1, \ldots, a_p, \neg b_1, \ldots, \neg b_k$$

where $a_1, \ldots, a_p \in M_n^{\prec}$ and $b_1, \ldots, b_k, t \notin M_n^{\prec}$. If there is no such $\ell(n+1)$, then let $M_{n+1}^{\prec} = M_n^{\prec}$. Otherwise, let

$$M_{n+1}^{\prec} = cl_{mon}(M_n^{\prec} \cup \{cln(r_{\ell(n+1)})\}).$$

This given, we then have the following.

Theorem 5.9 *If P is a countable FC-normal program, then*

(i) $M^{\prec}$ is a stable model of P if $M^{\prec}$ is constructed via the Countable Normal Forward Chaining algorithm with respect to $\prec$, where $\prec$ is any *well-ordering of $nmon(P)$ of order type ω.*

(ii) (Completeness of the construction.) Every stable model of P is of the form $M^{\prec}$ for a suitably chosen well-ordering $\prec$ of $nmon(P)$ of order type ω where $P^{\prec}$ is constructed via the Countable Normal Forward Chaining algorithm.

FC-normal programs also possess what Reiter terms the "semi-monotonicity" property.

Theorem 5.10 *Let P_1 and P_2 be two FC-normal program such that $P_1 \subseteq P_2$ but $mon(P_1) = mon(P_2)$ (that is, P_1, P_2 have the same Horn part). Assume, in addition, that both are FC-normal with respect to the same consistency property. Then for every stable model M_1 of P_1, there is a stable model M_2 of P_2 such that*

(i) $M_1 \subseteq M_2$ and
(ii) $NG(M_1, P_1) \subseteq NG(M_2, P_2)$.

FC-normal programs also satisfy the *orthogonality of stable models* property with respect to their consistency property.

Theorem 5.11 *Let P be a FC-normal program with respect to a consistency property Con. Then if M_1 and M_2 are two distinct stable models of P, $M_1 \cup M_2 \notin Con$.*

We end this section with three more theorems which are analogues of results that hold for normal default theories.

Theorem 5.12 *Let P be a FC-normal program with respect to a consistency property Con. Suppose that $cl_{mon}\{cln(r) : r \in nmon(P)\}$ is in Con. Then P has a unique stable model.*

Theorem 5.13 *Suppose P is a FC-normal program and that $D \subseteq nmon(P)$. Suppose further that M_1' and M_2' are distinct stable models of $D \cup mon(P))$.*

Then P has distinct stable models M_1 and M_2 such that $M_1' \subseteq M_1$ and $M_2' \subseteq M_2$.

We say that $c \in U$ has a consistent proof scheme with respect to a consistency property Con over P iff there is a proof scheme

$$p = \langle\langle c_0, r_0, G_0\rangle, \dots, \langle c_m, r_m, G_m\rangle\rangle \tag{4}$$

such that $c_m = c$ and $\{c_0, \dots, c_m\} \in Con$. We then have the following.

Theorem 5.14 *Let P be a FC-normal program with respect to a consistency property Con. Then $c \in H_P$ is an element of some stable model of P iff c has a consistent proof scheme with respect to Con.*

6 An Application to Default Logic

First of all we show that in propositional default logic, default theories with a finite number of justification-free clauses are monotonically decidable.

Recall that a default theory $\langle D, W\rangle$ is a pair where D is a collection of default rules, that is, rules of form

$$\frac{\alpha\colon M\beta_1, \dots, M\beta_m}{\psi}, \tag{5}$$

(where $\alpha, \beta_1, \dots, \beta_m$, and ψ are formulas) and W a collection of formulas of the language $\mathcal{L}$.
We associate an operator, Γ mapping $\mathcal{P}(\mathcal{L})$ into $\mathcal{P}(\mathcal{L})$ by stipulating: $\Gamma(S) = T$ if T is the least theory in $\mathcal{L}$ such that $W \subseteq T$, T is closed under propositional consequence and T satisfies the following condition:

$$\text{If } d = \frac{\alpha\colon M\beta_1, \dots, M\beta_m}{\psi} \in D, \alpha \in T, \neg\beta_1 \notin S, \dots, \neg\beta_m \notin S, \text{ then } \psi \in T$$

Now, a theory $S \subseteq \mathcal{L}$ is called default extension of $\langle D, W\rangle$ if $\Gamma(S) = S$.

Represent a default theory as a program consisting of three lists:
(i) Elements $\gamma \in W$ are represented as clauses:

$$\gamma \leftarrow$$

(ii) Rules of form (5) are represented as clauses

$$\gamma \leftarrow \alpha, \mathbf{not}\neg\beta_1, \dots, \mathbf{not}\neg\beta_m$$

(That is, the restraints of the clause representing a default rule r have an additional negation-as-failure symbol in front).

(iii) Processing rules of logic. That is, all the monotonic rules of the system of classical logic.

We then have the following proposition from [MNR90]:

Proposition 6.1 *A collection $S \subseteq \mathcal{L}$ is a stable model of a program consisting of clauses of type* (i), (ii), *and* (iii) *if and only if S is a default extension of $\langle D, W \rangle$.*

We assume that the propositional language $\mathcal{L}$ is effectively enumerated (that is its atoms form an r.e. set)

Proposition 6.2 *Let $\langle D, W \rangle$ be a default theory with a finite number of justification-free rules and finite W. Then the program P corresponding to $\langle D, W \rangle$ is monotonically decidable.*

Proof: Notice that under our translation $mon(P)$ is infinite and consists of these clauses:

(i) $\varphi \leftarrow$ for φ a tautology;
(ii) $\psi \leftarrow \varphi, \varphi \supset \psi$ for $\varphi, \psi \in \mathcal{L}$;
(iii) $\varphi \leftarrow \alpha_1, \ldots, \alpha_n$ where $\frac{\alpha_1, \ldots, \alpha_n :}{\varphi}$ is a justification-free rule in D.
(iv) $\gamma \leftarrow$ for $\gamma \in W$.

Now, given a finite A, we can construct $cl_{\text{mon}}(A)$ in stages as follows.

Stage 0. Let $W_0 = A \cup W$
Stage $s+1$. Assume we have constructed W_s. Let I_{s+1} be the set of all φ such that there is a clause

$$\varphi \leftarrow \alpha_1, \ldots, \alpha_n$$

of the form (3) such that $\varphi \notin W_s$ and for all $i \leq n$

$$\left(\bigwedge_{\omega \in W_s} \omega \right) \supset \alpha_i$$

is a tautology.

If $I_{s+1} = \emptyset$, then set $W_{s+1} = W_s$ and stop. Otherwise let $W_{s+1} = W_s \cup I_{s+1}$.

It is easy to see that since there are only finitely many clauses in (3), there will be a stage s_0 such that $W_{s_0+1} = W_{s_0}$. Moreover since we are working in propositional logic, each stage is completely effective so that we can effectively compute s_0 and the corresponding finite set W_{s_0}. Then clearly

$$cl_{\text{mon}}(A) = \{\alpha : \left(\bigwedge_{\omega \in W_{s_0}} \omega \right) \supset \alpha\}$$

is a recursive set and our procedure shows that that there exists a recursive function f such that if $D_k = A$, then $\varphi_{f(k)}$ is a characteristic function of the set $cl_{\text{mon}}(A)$. □

Proposition 6.2 and Proposition 4.12 immediately imply the following corollary.

Corollary 6.3 *If $\langle D, W\rangle$ is a finite propositional default theory, then every extension of $\langle D, W\rangle$ is recursive.*

Moreover, we can translate the construction of $D^{\prec}$ to the context of default logic (regardless if D or W are finite or not). This reverse translation, from a program to a default theory produces, out of a well-ordering $\prec$ of the non-monotonic part of D, a subset D_1 and a theory T such that T is an extension of $\langle D_1, W\rangle$.

Thus, even when $\langle D, W\rangle$ has no extension we can still effectively extract a meaningful part out of $\langle D, W\rangle$.

7 Forward Chaining and Stratification

In this section we investigate our Forward Chaining method for stratified programs. Following Apt, Blair and Walker [ABW87] and Przymusinski [Prz87], we call a program P (locally) *stratified* if there exists an ordinal ν and a function $rank : U \to \nu$ such that for every clause $r = c \leftarrow a_1, \ldots, a_n, \neg b_1, \ldots, \neg b_m$, $rank(a_i) \leq rank(c)$, for all i, $1 \leq i \leq n$ and $rank(b_i) < rank(c)$, for all i, $1 \leq i \leq m$. The ordinal ν is called *the length of the stratification rank.*

Using a generally well-known argument (see Marek and Truszczyński [MT93], Section 6.7, for complete presentation) one can show that a stratified program possesses a unique stable model.

We next consider stable submodels of stratified programs. First let us look at an example.

Example 7.1 Let $H_P = \{a, b, c\}$, $P = \{r_1, r_2\}$, where

$$r_1 = a \leftarrow \neg b \qquad\qquad r_2 = b \leftarrow \neg c$$

This is a stratified program. The rank function is given by:

$$rank(c) = 0, \quad rank(b) = 1, \quad rank(a) = 2$$

The unique stable model of P is $\{b\}$. Notice that $mon(P) = \emptyset$. Consider now the ordering $\prec_1$ given by

$$r_2 \prec_1 r_1$$

With the ordering $\prec_1$ our algorithm computes the stable submodel $\{b\}$ which is, as noticed above, the unique stable model of P. Now consider the ordering $\prec_2$ given by

$$r_1 \prec_2 r_2$$

Now, the stable submodel computed with $\prec_2$ is $\{a\}$. Moreover the clause r_2 is inconsistent. □

Looking at this example we realize that the conclusion of the clause r_1 has the rank greater than that of the clause r_2, yet the clause r_1 was put earlier in the ordering $\prec_2$. When such an anomaly is eliminated, the stable submodels produced by such ordering will always be a stable model. That is, we say that a well-ordering $\prec$ of *nmon*(P) is *consistent with the stratification rank* if for every pair of clauses $r, s \in nmon(P)$, $r \prec s$ implies $rank(c(r)) \leq rank(c(s))$. That is the nonmonotonic clauses of P are sorted according to the rank of their conclusions (but if these conclusions have the same rank, then the ordering is arbitrary). We use the notion of ordering consistent with stratification to prove the basic result of this section.

Theorem 7.1 *Let P be a stratified logic program with a stratification rank and let $\prec$ be a well-ordering of nmon(P) consistent with stratification rank. Then the stable submodel of P generated by $\prec$ is the perfect model of P.*

The proof of Theorem 7.1 requires a series of lemmas.

Lemma 7.2 *Let P be a stratified program with stratification rank and let $\prec$ be a well-ordering of nmon(P) consistent with rank. If $Z \subseteq H_P$ then for all $x \in cl_{mon}(Z \cup \{c\}) \setminus cl_{mon}(Z)$, $rank(x) \geq rank(c)$.*

Proof: By induction on rank of c, using definition of stratified programs. □

Next we have.

Lemma 7.3 *Let P be a stratified program with stratification rank and let $\prec$ be a well-ordering of nmon(P) consistent with rank. Then for any λ, if r_λ is defined at stage λ of the FC-construction with respect to $\prec$, then for all $\delta > \lambda$ such that r_δ is defined, $rank(c(r_\delta)) \geq rank(c(r_\lambda))$.*

Proof: First we make the following observation. Suppose that $\lambda = \rho + 1$ and that

$$r_\lambda = z \leftarrow a_1, \dots, a_n, \neg b_1, \dots, \neg b_m.$$

Thus $z \notin D^{\prec}_{\rho}$, $a_1, \dots, a_n \in D^{\prec}_{\rho}$, $cl_{\text{mon}}(D^{\prec}_{\rho} \cup \{z\}) \cap (R^{\prec}_{\rho} \cup \{b_1, \dots, b_m\}) = \emptyset$. Then it is easy to show that any element $x \in cl_{\text{mon}}(D^{\prec}_{\rho} \cup \{z\}) \setminus D^{\prec}_{\rho}$ must have $rank(x) \geq rank(z)$. That is, suppose $x \in cl_{\text{mon}}(D^{\prec}_{\rho} \cup \{z\}) \setminus D^{\prec}_{\rho}$. Then if $x \neq z$, there must be a sequence of Horn clauses $\langle r_0, \dots, r_k \rangle$ of the form

$$r_i = c_i \leftarrow a^i_0, \dots, a^i_{k_i}$$

such that

(i) $c_k = x$
(ii) for all $1 \leq i \leq k$, either $a^i_j \in D^{\prec}_{\rho}$, $a^i_j = z$ or $a^i_j = c_l$ where $l < i$ for each $j \leq k_i$, and
(iii) for all $1 \leq i \leq k$, $c_i \in cl_{\text{mon}}(D^{\prec}_{\rho} \cup \{z\}) \setminus (D^{\prec}_{\rho} \cup \{z\})$.

Note that since $c_i \in cl_{\text{mon}}(D^{\prec}_{\rho} \cup \{z\}) \setminus (D^{\prec}_{\rho} \cup \{z\})$ it must be the case that

$$\{a^i_0, \dots, a^i_{k_i}\} \cap \{z, c_0, \dots, c_{i-1}\} \neq \emptyset$$

since otherwise $\{a^i_0, \dots, a^i_{k_i}\} \subseteq D^{\prec}_{\rho}$ and hence $c_i \in cl_{\text{mon}}(D^{\prec}_{\rho}) = D^{\prec}_{\rho}$. Thus in particular $z \in \{a^0_0, \dots, a^0_{k_0}\}$ and hence $rank(c_0) \geq rank(z)$. Then if we assume by induction on j that $rank(c_j) \geq rank(z)$ for all $j < i$, then $rank(c_i) \geq \max(\{rank(a^i_0), \dots, rank(a^i_{k_i})\}) \geq \min(\{rank(z), rank(c_0), \dots, rank(c_{i-1})\})$ $\geq rank(z)$. Thus $rank(x) \geq rank(z)$. This in turn implies that $D^{\prec}_{\rho+1} \setminus D^{\prec}_{\rho} \subseteq \{x : rank(x) \geq rank(c(r_\lambda))\}$.

Now suppose that there is a $\delta > \lambda$ such that r_δ is defined and

$$rank(c(r_\delta)) < rank(c(r_\lambda)).$$

Then pick δ as small as possible and let $\mu + 1 = \delta$. Thus for all $\lambda \leq \gamma \leq \mu$, $rank(c(r_\gamma)) \geq rank(c(r_\lambda))$. Hence using the observation above, it is easy to prove by induction on γ that $D^{\prec}_{\gamma} \setminus D^{\prec}_{\rho} \subseteq \{x : rank(x) \geq rank(c(r_\lambda))\}$. Hence $D^{\prec}_{\mu} \setminus D^{\prec}_{\rho} \subseteq \{x : rank(x) \geq rank(c(r_\lambda))\}$. But now consider

$$r_\delta = z' \leftarrow a'_1, \dots, a'_n \neg b'_1, \dots, \neg b'_m.$$

Thus $z' \notin D^{\prec}_{\mu}$, $a_1, \dots, a_n \in D^{\prec}_{\mu}$, $cl_{\text{mon}}(D^{\prec}_{\mu} \cup \{z'\}) \cap (R^{\prec}_{\mu} \cup \{b'_1, \dots, b'_m\}) = \emptyset$. But then for each i, $rank(a'_i) \leq rank(z') < rank(c(r_\lambda))$ so that it must be the case that $a'_i \in D_\rho$ since $D^{\prec}_{\mu} \setminus D^{\prec}_{\rho} \subseteq \{x : rank(x) \geq rank(c(r_\lambda))\}$. Moreover, since $cl_{\text{mon}}(D^{\prec}_{\mu} \cup \{z'\}) \cap (R^{\prec}_{\mu} \cup \{b'_1, \dots, b'_m\}) = \emptyset$, we certainly have $cl_{\text{mon}}(D^{\prec}_{\rho} \cup \{z'\}) \cap (R^{\prec}_{\rho} \cup \{b'_1, \dots, b'_m\}) = \emptyset$. But this means that r_δ is a candidate to be r_λ at stage λ. But since $rank(c(r_\delta)) < rank(c(r_\lambda))$, r_δ precedes r_λ in our ordering of clauses which would violate the fact that r_λ is the least applicable clause at stage λ. Thus there can be no such δ. $\square$

Lemma 7.3 implies immediately the following corollary:

Corollary 7.4 *Let P be a stratified program with stratification rank and let $\prec$ be a well-ordering of $nmon(P)$ consistent with stratification rank. Then for any λ, if r_λ is defined at stage λ of the FC-construction with respect to $\prec$, and $rank(c(r_\lambda)) = \xi$, then $R^{\prec}_{\lambda} \subseteq \{x : rank(x) < \xi\}$.*

We next have the following lemma.

Lemma 7.5 *Let P be a stratified program. If an atom $y \in H_P$ possesses a derivation from a set $Z \subseteq H_\rho$ using only the clauses in $mon(P)$, then y possesses such derivation from the set $Z \cap \{x : rank(x) \leq rank(y)\}$.*

Proof: We proceed by induction on the length of derivation. If y is a conclusion of an axiom, then y possesses a derivation from empty set, $\emptyset$, which is identical with $\emptyset \cap \{x : rank(x) \leq rank(y)\}$.

For the inductive step, note that if y is a conclusion of a Horn clause

$$r = y \leftarrow a_1, \ldots, a_m,$$

then by since P is stratified, $rank(a_1), \ldots rank(a_m) \leq rank(y)$. But then each of the a_i has a derivation shorter than the derivation of y and hence by induction, each a_i has a derivation from $Z \cap \{x : rank(x) \leq rank(a_i)\}$ and thus from $Z \cap \{x : rank(x) \leq rank(y)\}$. Combining these derivations together with r. we get the desired result. $\square$

Corollary 7.6 *If P is a stratified logic program with stratification rank and $y \in cl_{mon}(Z)$, then $y \in cl_{mon}(Z \cap \{x : rank(x) \leq rank(y)\})$.*

Next, Lemma 7.5 is used to prove the following:

Lemma 7.7 *Let P be a stratified program with stratification rank and let $\prec$ be a well-ordering of nmon(P) consistent with stratification rank. Then for any α, if r_α is defined at stage α of the FC-construction with respect to $\prec$ and $rank(c(r_\alpha)) = \xi$, then*

$$D_\alpha^\prec \cap \{x : rank(x) < \xi\} = (\bigcup_{\beta<\alpha} D_\beta^\prec) \cap \{x : rank(x) < \xi\}$$

Proof: Clearly, α is not a limit ordinal since if α is a limit ordinal, r_α is not defined. So let $\alpha = \gamma + 1$. The inclusion $\supseteq$ is immediate. To prove the inclusion $\subseteq$ recall that

$$D_\alpha^\prec = cl_{\mathrm{mon}}(D_\gamma^\prec \cup \{c(r_\alpha)\}) \tag{6}$$

If $y \in D_\alpha^\prec$, $rank(y) < \xi$, then there is a derivation of y from $D_\gamma^\prec \cup \{c(r_\alpha)\}$ that uses Horn clauses only. But since our system is stratified, there is a derivation of y from $(D_\gamma^\prec \cup \{c(r_\alpha)\}) \cap \{x : rank(x) \leq rank(z)\} = D_\gamma^\prec \cap \{x : rank(x) \leq rank(z)\}$ (Lemma 7.5). This implies that y belongs to the right hand side of (6). $\square$

Next, we characterize $D_\xi^\prec$ as the closure of a set of conclusions of clauses in $nmon(P)$ by means of Horn clauses.

Lemma 7.8 *Let P be a stratified logic program with stratification rank and let $\prec$ be a well-ordering of nmon(P) consistent with stratification rank. Then for any α,*

$$D_\alpha^\prec = cl_{mon}(\{c(r_\eta) : \eta \leq \alpha \wedge \ r_\eta \ \textit{is defined}\})$$

Proof: Only the case when r_α is defined needs proof since if α is a limit ordinal, the result is obvious, and if α is a nonlimit ordinal where r_α is not defined, the construction stopped.

So, assuming that r_α is defined, then $\alpha = \beta + 1$ and

$$D_\alpha^{\prec} = cl_{\mathrm{mon}}(D_\beta^{\prec} \cup \{c(r_\alpha\})$$

Now using the inductive assumption and the fact that $cl_{\mathrm{mon}}(\cdot)$ is monotonic and idempotent we get the desired conclusion. □

Now recall the construction of the unique stable model of a stratified logic program as presented in [MT93]. Define

$$nmon_\xi = \{r : r \in nmon(P) \wedge rank(c(r)) = \xi\}$$

$$mon_\xi = \{r : r \in mon(P) \wedge rank(c(r)) = \xi\}$$

We also define

$$nmon_{\leq\xi} = \bigcup_{\eta\leq\xi} nmon_\eta \qquad mon_{\leq\xi} = \bigcup_{\eta\leq\xi} mon_\eta$$

$$nmon_{<\xi} = \bigcup_{\eta<\xi} nmon_\eta \qquad mon_{<\xi} = \bigcup_{\eta<\xi} mon_\eta$$

Let us now look at the construction of [MT93], Section 6.7. This is, essentially, the construction of [ABW87] extended to transfinite. One constructs a family of subsets of H_P, $\langle M_\xi \rangle_{\xi<\nu}$ as follows:

(i) Since P is stratified, $nmon_0 = \emptyset$ and M_0 is defined as the closure of $\emptyset$ under the clauses in mon_0.

(ii) If M_η, $\eta < \xi$ is defined, we put $M_{<\xi} = \bigcup_{\eta<\xi} M_\xi$. Next, we *reduce* the clauses in $nmon_\xi$ by $M_{<\xi}$. That is, for each clause

$$C = c \leftarrow a_1, \dots, a_n \neg b_1, \dots, \neg b_m$$

in $nmon_\xi$, if for some i, $1 \leq i \leq m$, $b_i \in M_{<\xi}$, then this clause is eliminated. In the remaining clauses the negative parts are eliminated. In this fashion we get a set Q_ξ of Horn clauses. Then M_ξ is defined as the closure of $M_{<\xi}$ under clauses in $mon_\xi \cup Q_\xi$.

(iii) Finally we set $M = \bigcup_{\xi<\nu} M_\xi$ (recall that ν is the length of stratification). It has been proved in [MT93] that M is the unique stable model of P.

Since all the clauses in $mon_\xi \cup Q_\xi$ have the conclusion of the rank precisely ξ it follows that for every $\xi < \nu$,

$$M \cap \{x : rank(x) \leq \xi\} = M_\xi$$

Define now $p(\xi)$ as the least ordinal greater or equal than all α such that r_α is defined and $rank(r_\alpha) \leq \xi$. Clearly, function $p(\cdot)$ is well defined. Lemma 7.3 implies the following.

Lemma 7.9 *Function $p(\cdot)$ is (weakly) monotonic, that is*

$$\xi_1 < \xi_2 \textit{ implies } p(\xi_1) \leq p(\xi_2)$$

We shall prove now the crucial lemma in the proof of Theorem 7.1.

Lemma 7.10 *Let P be a stratified logic program with stratification rank and let $\prec$ be a well-ordering of nmon(P) consistent with stratification rank. Then for any ξ smaller than the length of stratification rank*

$$D^{\prec}_{p(\xi)} \cap \{x : rank(x) \leq \xi\} = M_\xi. \tag{7}$$

Proof: We proceed by induction on ξ. If $\xi = 0$, then the left hand side of 7 is the closure of $\emptyset$ under $mon(P)$ whereas the right hand side is the closure of $\emptyset$ under the clauses in mon_0. The desired equality then follows from Lemma 7.5.

Now assume that for all $\eta < \xi$,

$$D^{\prec}_{p(\eta)} \cap \{x : rank(x) \leq \eta\} = M_\eta.$$

We prove that $D^{\prec}_{p(\xi)} \cap \{x : rank(x) \leq \xi\} = M_\xi$. To this end we show the inclusion of the left hand side in the right hand side and conversely. For the inclusion $\subseteq$, we proceed by induction on ordinals η such that $rank(c(r_\eta)) \leq \xi$. The base step is very similar to the base step of outer induction and we leave it to the reader. The limit step is obvious.

Now assume that $\alpha = \gamma + 1$ and that $D^{\prec}_{\gamma} \cap \{x : rank(x) \leq \xi\} \subseteq M_\xi$. Consider the clause r_α. By Corollary 7.4, the elements of negative part of the body of of r_α all have the rank smaller than ξ. It is easy to see that $cons(r_\alpha) \cap M_{<\xi} = \emptyset$ since the negative literals of r_α are all of rank smaller than ξ and do not belong to $cl_{\text{mon}}(D^{\prec}_{\gamma})$ and by the (outer) induction hypothesis $D^{\prec}_{\gamma}$ contains all M_η for $\eta < \xi$. This implies that the Horn clause $c(r_\alpha) \leftarrow prem(r_\alpha)$ belongs to Q_ξ. Since $prem(r_\alpha) \subseteq D^{\prec}_{\gamma}$, by the (inner) induction hypothesis, $prem(r_\alpha) \subseteq M_\xi$. Since M_ξ is closed under the clauses of Q_ξ, $c(r_\alpha) \in M_\xi$. This, in turn implies that the set of $c(r_\eta)$ for all $\eta \leq \alpha$ for nonlimit ordinals is included in M. Indeed, by the inner inductive assumption, the conclusions of all clauses r_η with $\eta < \alpha$ belong to M_ξ and thus to M as well. We can now use Lemma 7.8 to prove that since M is closed under all Horn clauses in N, $D^{\prec}_{\alpha}$ is entirely included in M. But then

$$D^{\prec}_{\alpha} \cap \{x : rank(x) \leq \xi\} \subseteq M \cap \{x : rank(x) \leq \xi\} = M_\xi$$

This completes the (inner) induction argument for the inclusion $\subseteq$.

We now show the inclusion $\supseteq$. Clearly, we only need to prove that whenever $z \in M_\xi$ and $rank(z) = \xi$, then $z \in D_{p(\xi)}$. (for z's of smaller rank the inductive assumption immediately implies the result). But if $z \in M_\xi$ and $rank(z) = \xi$, then z possesses a (monotonic) derivation from $\bigcup_{\eta<\xi} M_\eta$ using the clauses in $mon_\xi \cup Q_\xi$. By induction on the length of such derivation, we prove that $z \in D^{\prec}_{p(\xi)}$. Notice that whenever $d = y \leftarrow a_1, \ldots, a_m$ is a clause in Q_ξ, then $rank(y) = \xi$. Also for some $b_1, \ldots b_n \notin M_\xi$, $r_d = y \leftarrow a_1, \ldots, a_m \neg b_1, \ldots, \neg b_n$ belongs to N. But then the rank of each $b_1, \ldots, b_n$ is strictly smaller than ξ. Since $b_1, \ldots, b_n \notin M_{<\xi}$, it follows that for each $\eta < p(\xi)$, $b_1, \ldots, b_n \notin D^{\prec}_\eta$. Indeed,

$$M_{<\xi} = D^{\prec}_{\bigcup_{\eta<\xi} p(\eta)} \cap \{x : rank(x) < \xi\}$$

and at stages above $\bigcup_{\eta<\xi} p(\eta)$, no element of rank smaller than ξ is added to $D^{\prec}$.

Now suppose $z \in M_\xi$, $rank(z) = \xi$, and z has a derivation of length 1 from $M_{<\xi}$ using the clauses from $mon_\xi \cup Q_\xi$. Then either z is a conclusion of an axiom in mon_ξ, in which case $z \in D^{\prec}_0$, or else there is an axiom d in Q_ξ with conclusion z. In that case there is a clause

$$r_d = z \leftarrow \neg b_1, \ldots, \neg b_n$$

in N, $b_1, \ldots, b_n \notin D^{\prec}_{p(\xi)}$. The clause r_d is, thus, always applicable. Eventually by some stage ρ the clause r_d becomes the first clause which can be applied if z is not already in $D^{\prec}_\rho$. Thus $z \in D^{\prec}_{p(\xi)}$.

In the inductive step we reason similarly. If z possesses a derivation from $M_{<\xi}$ of length $k+1$ using the clauses from $mon_\xi \cup Q_\xi$, then there is a clause $d = z \leftarrow a_1, \ldots, a_m$ belonging to $mon_\xi \cup Q_\xi$ used in this derivation. Since $a_1, \ldots, a_m$ have derivations of length at most k, they belong to $D^{\prec}_{p(\xi)}$ by our inductive assumption. Thus for some $\eta < p(\xi)$, all $a_1, \ldots, a_m$ belong to $D^{\prec}_\eta$. As in the base case of our induction, we need to consider two cases. If $d \in mon_\xi$, then $z \in D^{\prec}_\eta$ since the latter set is closed under all Horn clauses. If $d \in Q_\xi$, then for some $b_1, \ldots, b_m$ all of which do not belong to $D^{\prec}_{p(\xi)}$ the clause

$$r_d = z \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_n$$

belongs to N. The clause r_d is, therefore, applicable starting at η (if z is not already in $D^{\prec}_\eta$). Thus it, eventually, at some stage ρ, r_d will becomes the first applicable clause (again, if z is not derived earlier). Therefore, definitely, $z \in D^{\prec}_{p(\xi)}$. This completes the proof of inclusion $\supseteq$ and of the lemma. $\Box$

Proof of Theorem 7.1: Let ν be the length of stratification. We need to prove that $D^{\prec} = M$. But

$$D^{\prec} = \bigcup_{\eta} D^{\prec}_\eta = \bigcup_{\xi<\nu} D^{\prec}_{p(\xi)}$$

since the hierarchy of sets $D^{\prec}_{p(\xi)}$ is increasing and cofinal in the hierarchy of sets $D^{\prec}_{\eta}$. By Lemma 7.10

$$\bigcup_{\xi<\nu} D^{\prec}_{p(\xi)} = \bigcup_{\xi<\nu} M_{\xi} = M.$$

Thus $D^{\prec} = M$ as claimed. □

8 Modifications of the Construction

We will briefly discuss several modifications of the forward chaining construction described in our paper. To this end we need a short introduction to various three-valued interpretations. A general survey of these constructions (and their generalizations, for instance in bilattice setting) can be found in [Fi98].

A three-valued interpretation of a program is a pair $I = \langle T, F\rangle$ of sets of atoms so that $T \cap F = \emptyset$. Such interpretation assigns to an atom a truth value from the set $\{0, \perp, 1\}$. Namely, $I(a) = 1$ if $a \in T$, $I(a) = 0$ if $a \in F$ and $I(a) = \perp$ otherwise. Truth value can be easily extended to literals, by defining $\neg 0 = 1$, $\neg 1 = 0$, and $\neg\perp = \perp$.

Given a propositional program P (or a ground version of a predicate program) we can assign to it various operators in the space of interpretations.

The Kunen-Fitting operator (called "Kripke-Kleene" in [Fi98]) assigns to an interpretation I an interpretation I' as follows:

(i) $I'(p) = 1$ if for some clause $C = p \leftarrow l_1, \ldots, l_n$ in P, $I(l_1) = \ldots = I(l_n) = 1$
(ii) $I'(p) = 0$ if for every clause $C = p \leftarrow l_1, \ldots, l_n$ in P, for some $j \leq n$, $I(l_j) = 0$
(iii) $I'(p) = \perp$ otherwise.

This operator possesses a least fixpoint. This fixpoint is a three-valued model of the program.

Van Gelder, Ross and Schlipf [VGRS91] introduced another operator, leading to other three-valued model of a logic program. It is based on the notion of *unfounded set*. Given a three-valued interpretation I, an unfounded set with respect to I is any set of atoms X with the following property:

- Whenever $p \in X$, then for every clause $C = p \leftarrow l_1, \ldots, l_n$ in program P, for some $j \leq n$ either $I(l_j) = 0$ or l_j is an atom which belongs to X (*or* is not exclusive here).

There is always a largest unfounded set with respect to any interpretation I. Now define a new interpretation I' as follows. $I'(p) = 1$ if for some clause

$C \in P$, $C = p \leftarrow l_1, \ldots, l_n$, $I(l_1) = \ldots = I(l_n) = 1$. $I'(p) = 0$ if p belongs to the largest unfounded set with respect to I. Finally, $I'(p) = \perp$ for the remaining atoms p. It is clear that we defined a three-valued interpretation. The operator assigning I' to I can be iterated and it also possesses the least fixpoint. This fixpoint is called well-founded model of P.

Well-founded model generalizes stable model, in the sense that if M is a stable model of P than the interpretation $\langle M, At \setminus M\rangle$ is a fixpoint of the operator described above. Moreover, well-founded model approximates stable models. That is positive part of well-founded models is included in the intersection of all stable models, whereas the negative part of it is included in the intersection of the complements of stable models. Fitting's paper [Fi98] contains an extensive discussion of abstract treatment of well-founded semantics and its generalizations.

Since $D^{\prec}_{\xi} \cap R^{\prec}_{\xi} = \emptyset$, the pair $\langle D^{\prec}_{\xi}, R^{\prec}_{\xi}\rangle$ is a three-valued interpretation. In particular $\langle D^{\prec}, R^{\prec}\rangle$ is a three-valued interpretation. It is natural to ask about the relationship of this interpretation to the well-founded interpretation. Observe that $\langle D^{\prec}, R^{\prec}\rangle$ does not need to be a three-valued model of the program. This happens when there are inconsistent clauses.

On the other hand, the well-founded interpretation is not always included in $\langle D^{\prec}, R^{\prec}\rangle$. Indeed, for a Horn program $P = \{p \leftarrow q, q \leftarrow p\}$ the interpretation $\langle D^{\prec}, R^{\prec}\rangle = \langle\emptyset, \emptyset\rangle$, whereas the well-founded interpretation fails both p and q.

It should be clear that the construction of $\langle D^{\prec}, R^{\prec}\rangle$ admits various modifications. Notice that we can increase at each stage ξ of the construction the negative side of the construction. Specifically, at each stage of the construction we can modify the set $R^{\prec}_{\xi}$ extending it to a larger set $S^{\prec}_{\xi}$ (as long as $D^{\prec}_{\xi} \cap S^{\prec}_{\xi} = \emptyset$) and use $S^{\prec}_{\xi}$ instead of $R^{\prec}_{\xi}$ in the later stages of the construction. What is the effect of such modifications? The result is that more clauses may become inconsistent and some clauses that could be applied in the Forward Chaining construction may become inapplicable. But the basic result, namely that $D^{\prec}$ is a stable model of $P \setminus I^{\prec}$ remains true. That is, some previously applicable clauses may become inapplicable or inconsistent, but when the latter are eliminated the constructed set is a stable model of the resulting program. The intervention is, however, drastic. That is, since some clauses previously applicable may become inapplicable, there is no natural relationship between the models obtained from the modified construction and those obtained from the original one.

Moreover, once we select a clause for application, we can close the sets $D^{\prec}_{\xi}$ and $R^{\prec}_{\xi}$ under various operators. Specifically we can apply Kunen-Fitting operator (with the iteration to ω or further) or Van Gelder, Ross, Schlipf construction. All such modifications are possible and lead to a cross over of our theory with other three-valued approaches.

Finally, notice that in the case of finite propositional programs (or finite predicate programs without function symbols) all these constructions can be performed in polynomial time. The "straightforward" Forward Chaining construction seems to be the simplest.

9 Conclusions

We introduced a novel technique for computing stable models of programs (and so, by interpretability results, also of default extensions, and answer sets for logic programs with classical negation). In contrast with other techniques of finding stable models our algorithm always computes a subset of the base of the program. Moreover, this subset is a stable model of a subprogram of the original program. We feel that the technique introduced in this paper will have applications in real-time systems for computing values of default statements and parameters. The class of stable submodels (which properly contains stable models) is interesting in its own right and deserves further study. For example, our Forward Chaining construction suggests a new semantics for logic programs and default theories. That is, given a program P, we say a stable submodel $D^{\prec}$ is *maximal* if there is no ordering $\prec'$ such that $A^{\prec} \subset A^{\prec'}$. That is the set of inconsistent clauses is minimal (recall that $A^{\prec} = P \setminus I^{\prec}$). Note that if P has a stable model M, then there is an ordering $\prec'$ such that $D^{\prec'} = M$ and so $A^{\prec'} = P$. Thus every maximal stable submodel must also have $A^{\prec} = P$ and so, in this case, every maximal stable submodel is in fact an stable model. Thus if P has a stable model then the set of maximal stable submodels, $\mathcal{M}SS(P)$ is just the set of stable models. However $\mathcal{M}SS(P)$ is nonempty for all programs. Thus the set of maximal stable submodels extends the usual stable semantics. We shall explore the properties of maximal stable submodels in later papers.

References

[AvE82] K.R. Apt and M.H. van Emden. Contributions too the theory of logic programming. *Journal of the ACM*, 29:841–862, 1982

[ABW87] K. Apt, H.A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–142, Los Altos, CA, 1987. Morgan Kaufmann.

[Apt90] K. Apt. Logic programming. In J. van Leeuven, editor, *Handbook of Theoretical Computer Science*, pages 493–574. Cambridge, MA, 1990, MIT Press.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 293–322. Plenum Press, 1978.

[DG84] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.o

[Doy79] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[Fi85] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2:295-312, 1985.

[Fi98] M. Fitting. Fixpoint Semantics for Logic Programming, A survey Proceedings of the 12th Workshop on Mathematical Foundations of Programming Semantics. Special issue of Theoretical Computer Science, to appear.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Symposium on Logic Programming*, pages 1070–1080, Cambridge, MA., 1988. MIT Press.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *Logic Programming: Proceedings of the 7th International Conference*, pages 579–597, Cambridge, MA., 1990. MIT Press.

[GS92] J. Grant and V.S. Subrahmanian. Reasoning about inconsistent knowledge bases. *IEEE Trans. on Knowledge and Data Engineering*, to appear.

[GS93] J. Grant and V.S. Subrahmanian. The optimistic and cautious semantics for inconsistent knowledge bases. Department of Computer Science, University of Maryland, 1993.

[JS72a] C.G. Jockusch and R.I. Soare. Degrees of members of π_1^0 classes. *Pacific Journal of Mathematics*, 40:605–616, 1972.

[JS72b] C.G. Jockusch and R.I. Soare. π_1^0 classes and degrees of theories. *Transactions of American Mathematical Society*, 173:33–56, 1972.

[KL89] M. Kifer and E. Lozinskii. RI: A logic for reasoning about inconsistency. TARK IV, Asilomar, CA, pages 253-262, 1989.

[KN93a] W. Kohn and A. Nerode. Models for Hybrid Systems: Automata, Topologies, Controllability, Observability. In: *Hybrid Systems*, R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel, eds. Springer Lecture Notes in Computer Science 736, pages 317-356, 1993.

[Ku87] K. Kunen. Negation in Logic Programming. *Journal of Logic Programming*, 4:289-308, 1987.

[MNR90] W. Marek, A. Nerode, and J.B. Remmel. Nonmonotonic rule systems I. *Annals of Mathematics and Artificial Intelligence*, 1:241–273, 1990.

[MNR92c] W. Marek, A. Nerode, and J.B. Remmel. Nonmonotonic rule systems II. *Annals of Mathematics and Artificial Intelligence*, 5:229–263, 1992.

[MNR92a] W. Marek, A. Nerode, and J. B. Remmel. The stable models of predicate logic programs. In K.R. Apt, editor, *Proceedings of International Joint Conference and Symposium on Logic Programming*, pages 446–460, Boston, MA, 1992. MIT Press, to appear in *Journal of Logic Programming*.

[MNR95] W. Marek, A. Nerode, and J. B. Remmel. Complexity of Normal Default Logic and Related Modes of Nonmonotonic Reasoning, Proceedings of 10th Annual IEEE Symposium on Logic in Computer Science, pp. 178-187, 1995.

[MNR93b] W. Marek, A. Nerode, and J. B. Remmel. Context for Belief Revision: FC-Normal Nonmonotonic Rule Systems. Annals of Pure and Applied Logic 67(1994) pp. 269-324.

[MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38:588 – 619, 1991.

[MT93] W. Marek and M. Truszczyński. *Nonmonotonic Logic – Context-dependent reasonings* Berlin, Heidelberg, New York, 1993, Springer.

[McD82] D. McDermott. Nonmonotonic logic II: Nonmonotonic modal theories. *Journal of the ACM*, 29:33–57, 1982.

[MD80] D. McDermott and J. Doyle. Nonmonotonic logic I *Artificial Intelligence*, 13:41–72, 1980.

[Prz87] T. Przymusinski, On the declarative semantics of stratified deductive databases and logic programs, In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216, Los Altos, CA, 1987. Morgan Kaufmann.

[RDB89] M. Reinfrank, O. Dressler, and G. Brewka. On the relation between truth maintenance and non-monotonic logics. In *Proceedings of IJCAI-89*, pages 1206–1212, San Mateo, CA., 1989. Morgan Kaufmann.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Sco82] D. Scott. Domains for denotational semantics. In *Proceedings of ICALP-82*, pages 577–613, Heidelberg, 1982. Springer Verlag.

[VGRS91] A. Van Gelder, K.A. Ross and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. Journal of the ACM 38(1991).

[YBB92] F. Yang, H. Blair, and A. Brown. Programming in default logic. University of Syracuse, 1992.